Πανεπιστήμιο Δυτικής Αττικής

Σχολή Μηχανικών

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Σύγκριση της υλοποίησης εφαρμογών βάσεων δεδομένων με χρήση σχεσιακών (MySQL) και μη σχεσιακών Συστημάτων Διαχείρισης Βάσης Δεδομένων (MongoDB) και του Λογισμικού Διαχείρισης Περιεχομένου Wordpress.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

απο τους

ΝΙΚΟΛΑΟ ΚΑΡΑΠΙΠΕΡΗ και ΑΛΕΞΑΝΔΡΑ ΝΑΚΟΥ

Επιβλέπων: Χ. Σκουρλάς
Καθηγητής ΠΑΔΑ

Αθήνα, Ιανουάριος 2021

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΥΓΚΡΙΣΗ ΤΗΣ ΥΛΟΠΟΙΗΣΗΣ ΕΦΑΡΜΟΓΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΜΕ ΧΡΗΣΗ ΣΧΕΣΙΑΚΩΝ (MYSQL) ΚΑΙ ΜΗ ΣΧΕΣΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ (MONGODB) ΚΑΙ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ ΔΙΑΧΕΙΡΙΣΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ WORDPRESS.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

από τους

ΝΙΚΟΛΑΟ ΚΑΡΑΠΙΠΕΡΗ, ΑΛΕΞΑΝΔΡΑ ΝΑΚΟΥ

Επιβλέπων : Χρήστος Σκουρλάς
Καθηγητής ΠΑΔΑ.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Ιανουάριο του 2021.

Christos Skourlas Digitally signed by Christos Skourlas Date: 2021.07.07 08:37:43 +03'00'	KLEIO SGOUROPOULOU Digitally signed by KLEIO SGOUROPOULOU Date: 2021.07.07 13:23:18 +03'00'	Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2021.07.07 15:14:54 +03'00'
(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Χρήστος Σκουρλάς	Κλειώ Σγουροπούλου	Βασίλης Μάμαλης
Καθηγητής ΠΑΔΑ.	Καθηγήτρια ΠΑΔΑ.	Καθηγητής ΠΑΔΑ.

Αθήνα, Ιανουάριος 2021

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Καραπιπέρης Νικόλαος του Χαράλαμπου, με αριθμό μητρώου cs131003 και η κάτωθι υπογεγραμμένη Νάκου Αλεξάνδρα του Γεωργίου, με αριθμό μητρώου cs131059, φοιτητές του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνουμε υπεύθυνα ότι:

«Είμαστε συγγραφείς αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνουμε ότι αυτή η εργασία έχει συγγραφεί από εμάς αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μας, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μας ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μας».

Ο Δηλών

Η Δηλούσα

(Υπογραφή)

.....

ΝΙΚΟΛΑΟΣ ΚΑΡΑΠΙΠΕΡΗΣ

(Υπογραφή)

.....

ΑΛΕΞΑΝΔΡΑ ΝΑΚΟΥ

Copyright © Νικόλαος Καραπιπέρης, Αλεξάνδρα Νάκου 2021.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τους συγγραφείς και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Δυτικής Αττικής.

Περίληψη

Οι βάσεις δεδομένων είναι ένα αναπόσπαστο κομμάτι του σύγχρονου προγραμματισμού. Είναι απαραίτητες για την δημιουργία διαφόρων εφαρμογών που χρησιμοποιούνται κυρίως στον επαγγελματικό τομέα που είναι απαραίτητη η αποθήκευση και η χρήση πολυάριθμων δεδομένων. Στην συγκεκριμένη διπλωματική επιχειρήθηκε η δημιουργία τριών εφαρμογών ενοικίασης καταλυμάτων με χρήση των βάσεων MySQL και MongoDB, ούτως ώστε να εκτιμηθεί ποιά από αυτές τις βάσεις κρίνεται η καταλληλότερη για χρήση σε μία τέτοια εφαρμογή. Απευθύνεται κυρίως σε σπουδαστές πληροφορικής που μπορούν να κατανοήσουν σε βάθος τη διαδικασία υλοποίησης. Η υλοποίηση των εφαρμογών έγινε με χρήση των γλωσσών PHP και HTML, JavaScript, Node.js και HTML, ενώ χρησιμοποιήθηκε και το σύστημα διαχείρισης περιεχομένου Wordpress που δημιουργεί τη δική του βάση δεδομένων.

Λέξεις Κλειδιά: Βάσεις Δεδομένων, Συγκριτική Μελέτη, Διαδικτυακή Εφαρμογή, Airbnb, MySQL, MongoDB

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Databases are an integral part of modern programming. They are necessary for the creation of various applications that are used mainly in the professional field that require the storage and use of large amounts of data. In this dissertation, an attempt was made to create three accommodation rental applications using the MySQL and MongoDB databases in order to assess which of these databases is considered the most suitable for use in such an application. It is aimed mainly at computer science students who can understand the implementation process in depth. The implementation of the applications was done using the languages PHP plus HTML, JavaScript and Node.js plus HTML, while the Wordpress content management system was also used, which creates its own database.

Keywords: Databases, Comparative Study, Web Application, Airbnb, MySQL, MongoDB

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Δημιουργία Ιστοσελίδων Καταχώρισης, Εύρεσης και Ενοικίασης Καταλυμάτων για την Συγκριτική Μελέτη των Βάσεων MongoDB και MySQL

Μια βάση δεδομένων είναι μια οργανωμένη συλλογή δεδομένων. Είναι η συλλογή των σχημάτων, πινάκων, ερωτημάτων, εκθέσεων, προβολών και άλλων αντικειμένων. Χρησιμοποιούνται από παντός συστήματα διαχείρισης δεδομένων και πληροφοριών, και υποστηρίζουν σε απευθείας σύνδεση αλληλεπιδράσεις με τους πελάτες και τους προμηθευτές. Υπάρχουν πολλά διαφορετικά συστήματα βάσεων δεδομένων ωστόσο για την σύγκριση στην παρούσα εργασία επιλέχθηκαν η MySQL και η MongoDB. Η MySQL είναι ένα από τα πιο γνωστά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων ενώ η MongoDB είναι ανοιχτού κώδικα NoSQL βάση δεδομένων και ανήκει στην κατηγορία των document βάσεων δεδομένων.

Η πλατφόρμα Airbnb είναι μία δημοφιλής πλατφόρμα καταχώρισης, εύρεσης και ενοικίασης καταλυμάτων. Οι υπηρεσίες της πλατφόρμας δίνουν μία πληθώρα δεδομένων και λειτουργιών που καλείται να διαχειριστεί ο προγραμματιστής με τη βοήθεια της βάσης δεδομένων που θα σχεδιάσει και θα χρησιμοποιήσει.

1.2 Αντικέιμενο Διπλωματικής

Η παρούσα διπλωματική αποσκοπεί στην σύγκριση των βάσεων δεδομένων MySQL και MongoDB μέσα από τη δημιουργία τριών εφαρμογών τύπου airbnb με χρήση διαφορετικής τεχνικής στην καθεμιά. Μέσα από την δημιουργία την εφαρμογών γίνεται η μελέτη για τη λειτουργία των βάσεων και προκύπτουν συμπεράσματα ως προς τη χρήση τους με βάση την προγραμματιστική οπτική. Απευθύνεται κυρίως σε φοιτητές του αντικειμένου και παραθέτει σε αυτούς τα θετικά και τα αρνητικά της χρήσης κάθε βάσης όπως αυτή φάνηκε μέσα από την υλοποίηση των τριών εφαρμογών.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήσαμε τεχνοτροπίες και συστήματα βάσεων δεδομένων.
2. Μελετήσαμε τεχνικές κατασκευής ιστοσελίδων καταχώρισης, εύρεσης και ενοικίασης καταλυμάτων.
3. Υλοποιήσαμε τρεις ιστοσελίδες διαφορετικής τεχνοτροπίας τύπου airbnb.
4. Κατασκευάσαμε σενάρια χρήσης της κάθε πλατφόρμας.
5. Συγκρίναμε τα συστήματα βάσεων δεδομένων που χρησιμοποιήθηκαν.
6. Συμπεράναμε ποια τα θετικά και ποια τα αρνητικά στοιχεία των βάσεων.

1.3 Οργάνωση Κειμένου

Η διπλωματική εργασία απαρτίζεται συνολικά από οκτώ κεφάλαια. Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο κεφάλαιο δύο. Τεχνικές και μεθοδολογίες που χρησιμοποιήθηκαν περιέχονται στο τρίτο κεφάλαιο. Στο τέταρτο κεφάλαιο αναλύονται οι αρχιτεκτονικές και οι λειτουργίες που είναι απαραίτητες για να κατανοήσει κάποιος τις πλατφόρμες που υλοποιήθηκαν. Η αρχιτεκτονική της κάθε πλατφόρμας καθώς και παρουσίαση των βάσεων δεδομένων γίνεται στο πέμπτο κεφάλαιο ενώ η ανάλυση της υλοποίησης καθώς και ποιες πλατφόρμες και ποια προγραμματιστικά εργαλεία είναι απαραίτητα για να εκτελέσει κάποιος την εργασία παρουσιάζονται στο έκτο κεφάλαιο. Στο έβδομο κεφάλαιο δίνονται τα σενάρια χρήσης της κάθε πλατφόρμας και η παρουσίαση των αποτελεσμάτων των ελέγχων αυτών. Τέλος παρατίθενται τα συμπεράσματα μας και κάποιες ιδέες για μελλοντικές επεκτάσεις που μπορούν να βρεθούν στο κεφάλαιο οκτώ.

2 Σχετικές Εργασίες

Όπως αναφέρθηκε ήδη η εργασία βασίζεται στη δημιουργία πλατφορμών τύπου Airbnb βασισμένες σε τρεις διαφορετικές νοοτροπίες που κάνουν χρήση διαφορετικών τεχνοτροπιών αποθήκευσης δεδομένων και επεξεργασίας αυτών ούτως ώστε να μελετηθούν ως προς την ευκολία υλοποίησής, συντήρησης και απόδοσής τους. Η βασική πλατφόρμα που εξετάστηκε είναι η Airbnb καθώς είναι η πιο διαδεδομένη πλατφόρμα καταχώρισης, εύρεσης και ενοικίασης καταλυμάτων καθώς και παραπλήσιες αυτής, η Booking.com και η Trivago.

2.1 Πλατφόρμα Airbnb

Η πλατφόρμα Airbnb είναι η πιο γνωστή και διαδεδομένη πλατφόρμα αναζήτησης καταλύματος από επισκέπτες σε ολόκληρο τον κόσμο. Φιλοξενεί εκατομμύρια καταλύματα ενώ οποιοσδήποτε μπορεί να γίνει οικοδεσπότης εύκολα και γρήγορα. Ένα στοιχείο που κάνει την πλατφόρμα ιδανική για τον σκοπό μας και ταυτόχρονα αποτελεί και έναν από τους λόγους που είναι τόσο γνωστή στο ευρύ κοινό είναι η δυνατότητα search που παρέχει. Ο χρήστης μπορεί να ψάξει για καταλύματα με βάση το όνομα, τη διαθεσιμότητα, την περιοχή καθώς και τους πόσους επισκέπτες μπορεί να δεχθεί ένα οίκημα. Από προγραμματιστική άποψη η λειτουργία αυτή δίνει την δυνατότητα για πολλαπλή επικοινωνία με τη βάση. Η πλατφόρμα δίνει την δυνατότητα παροχής "εμπειριών" όπου χρήστες από όλων τον κόσμο μπορούν να διαφημίσουν και να πουλήσουν εμπειρίες σε άλλους χρήστες παρ' όλ' αυτά δεν κρίθηκε αναγκαία η χρήση της συγκεκριμένης υπηρεσίας για την ανάδειξή των λειτουργιών της βάσης οπότε και δεν εμφανίζεται στην διπλωματική μας.

2.2 Πλατφόρμα Booking.com

Η πλατφόρμα Booking.com είναι μία εξίσου γνωστή πλατφόρμα εύρεσης καταλύματος. Από τις βασικές διαφορές που παρατηρήθηκαν είναι ότι εδώ δίνονται περισσότερες επιλογές καθώς ο χρήστης μπορεί να βρει και τα εισιτήρια για το ταξίδι, καθώς και μεταφορικό μέσο για τη μεταφορά στο κατάλυμα του. Ωστόσο η δομή του είναι περισσότερο σύνθετη από τις ανάγκες μας και μέσα από σενάρια χρήσης συμπεράναμε ότι είναι παραπάνω πολύπλοκο για να παρουσιαστεί ένα σύστημα ανάλογο με αυτό.

2.3 Πλατφόρμα Trivago

Η Trivago είναι η γνωστή ιστοσελίδα σύγκρισης τιμών καταλυμάτων αλλά και μεταφορικών. Στην ουσία η πλατφόρμα αυτή χρησιμοποιεί δεδομένα που συλλέγει από τις υπόλοιπες πλατφόρμες φιλοξενίας και παρουσιάζει στον χρήστη την πιο οικονομική λύση για αυτόν. Βασική διαφορά με τις προηγούμενες πλατφόρμες είναι ότι η Trivago χρησιμοποιεί τη Redis βάση δεδομένων ενώ οι δύο πλατφόρμες που αναφέρθηκαν είναι χτισμένες με χρήση βάσης MySQL.

3 Θεωρητικό Υπόβαθρο

Στο παρόν κεφάλαιο καθώς και στα επιμέρους τμήματα αυτού θα παρουσιαστούν βασικές έννοιες και το θεωρητικό υπόβαθρο αυτών. Είναι σημαντικό να διασαφηνιστούν οι έννοιες αυτές σε αυτό το σημείο ούτως ώστε ο αναγνώστης να μπορέσει να κατανοήσει το αντικείμενο της πτυχιακής.

3.1 Βάσεις Δεδομένων

Σε αυτό το κεφάλαιο δίνεται μια συνοπτική εισαγωγή στις Βάσεις Δεδομένων και παρουσιάζονται τα διαφορετικά είδη αυτών με περαιτέρω ανάλυση στις MySQL και MongoDB.

Οι βάσεις δεδομένων καθώς και τα συστήματα αυτών αποτελούν ένα σημαντικό στοιχείο της καθημερινής ζωής στη σύγχρονη κοινωνία και εξασκούν σημαντική επίδραση στην αυξανόμενη χρήση των υπολογιστών. Βάση δεδομένων (database) είναι μια συλλογή από σχετιζόμενα δεδομένα. Με τον όρο δεδομένα εννοούμε γνωστά γεγονότα που μπορούν να καταγραφούν και που έχουν κάποια υπονοούμενη σημασία. Ο πιο πάνω ορισμός μιας βάσης δεδομένων είναι αρκετά γενικός ενώ η συνήθης χρήση του όρου βάση δεδομένων είναι αρκετά πιο περιορισμένη. Μια βάση δεδομένων έχει τις εξής ιδιότητες : αναπαριστά μία εικόνα του πραγματικού κόσμου που θα μπορούσε να αναφερθεί και ως μικρόκοσμος (miniworld) ή Πεδίο Αναφοράς (Universe of Discourse, UoD). Οποιαδήποτε αλλαγή στην εικόνα αυτή του πραγματικού κόσμου επηρεάζει άμεσα και την βάση δεδομένων καθώς αλλάζουν τα στοιχεία αυτής. Τα στοιχεία/ δεδομένα τα οποία αποθηκεύονται σε μία βάση δεδομένων θεωρείται ότι έχουν μία λογικά συνεκτική σημασία διαφορετικά δεν είναι σωστή η χρήση του όρου βάση δεδομένων για την περιγραφή αυτών. Τέλος μία συγκεκριμένη βάση δεδομένων δημιουργείται (σχεδιασμός, χτίσιμο και φόρτωση με δεδομένα) για ένα συγκεκριμένο σκοπό. Χρησιμοποιείται για μία προκαθορισμένη ενέργεια και στοχεύει μία συγκεκριμένη ομάδα ανθρώπων.

Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) (Database Management System - DBMS) είναι μια συλλογή από προγράμματα που επιτρέπουν στους χρήστες να δημιουργήσουν και να συντηρήσουν μια βάση δεδομένων. Είναι ουσιαστικά το σύστημα λογισμικού που χρησιμοποιείται για να οριστεί μία βάση δεδομένων , να μπορεί ο χρήστης να κάνει χρήση αυτής ενώ πραγματοποιείται έτσι και η δυνατότητα άλλες εφαρμογές να έχουν πρόσβαση στην ίδια αυτή βάση. Σημαντικό είναι να αναφερθεί ότι το ΣΔΒΔ παρέχει επίσης λειτουργίες για την προστασία της βάσης δεδομένων και τη συντήρηση αυτής για μακρύ χρονικό διάστημα. Ουσιαστικά μία εφαρμογή στέλνει ερωτήματα ή αιτήματα στο ΣΔΒΔ και έτσι γίνεται χρήση της βάσης δεδομένων για να απαντήθούν αυτά με βάση τα δεδομένα που αυτή περιέχει. Ανακτώνται τα δεδομένα-πληροφορίες που βρίσκονται αποθηκευμένα σε αυτή καθώς πραγματοποιείται ανάγνωση στη βάση και επιστρέφονται μέσω του ΣΔΒΔ στον χρήστη.

3.1.1 Σχεσιακές Βάσεις Δεδομένων

Το σχεσιακό μοντέλο παριστάνει τη βάση δεδομένων ως μια συλλογή από σχέσεις. Κάθε σχέση μοιάζει με έναν πίνακα, που περιέχει στοιχεία με βάση την κατηγορία την οποία παρουσιάζει ο πίνακας, ή με ένα "επίπεδο"αρχείο εγγραφών. Στην περίπτωση που δούμε την σχέση ως έναν πίνακα (table) τιμών τότε δεχόμαστε ότι κάθε γραμμή του πίνακα περιέχει μία εγγραφή που συσχετίζεται με τις υπόλοιπες γραμμές. Το σύνολο των εγγραφών αυτών αναφέρονται σε μία συγκεκριμένη οντότητα της βάσης η περιγραφή της οποίας χρησιμοποιείται ως όνομα του πίνακα βοηθητικά ούτως ώστε να γνωρίζουμε σε ποια οντότητα αναφέρονται τα συγκεκριμένα δεδομένα. Στο σχεσιακό μοντέλο κάθε γραμμή ενός πίνακα παριστάνει ένα γεγονός που αντιστοιχεί σε ένα γεγονός ή μία οντότητα του πραγματικού κόσμου. Σύμφωνα με την ορολογία ο πίνακας αναφέρεται ως σχέση, ο τίτλος κάθε στήλης ως γνώρισμα ενώ η γραμμή ως πλειάδα.

3.1.2 NoSQL Βάσεις Δεδομένων

Τα NoSQL (γνωστό και ως Not Only SQL) συστήματα και βάσεις δεδομένων αποιτελλούν μια ευρεία ομάδα συστημάτων διαχείρισης βάσεων δεδομένων (Database Management System) που το κύριο χαρακτηριστικό της είναι η μη τήρηση του μοντέλου RDBMS (Relational Database Management System), το οποίο και χρησιμοποιείται κατά κόρον από τα συστήματα της σύγχρονης εποχής. Οι NoSQL βάσεις δεδομένων γενικώς δεν χρησιμοποιούν κάποιο δομημένο σύστημα για τα στοιχεία που περιλαμβάνουν, όπως π.χ. πίνακες, ούτε χρησιμοποιούν κάποια Structured Query Language (SQL) για την διαχείριση των δεδομένων, αλλά χρησιμοποιούν αποκλειστικά non-relational τρόπους οργάνωσης και ανάλυσης των δεδομένων. Αυτό για το οποίο κατά κύριο λόγο είναι βέλτιστα στη χρήση τους τα NoSQL συστήματα είναι το να ανακτούν και να επισυνάπτουν δεδομένα. Παρουσιάζουν σημαντική μείωση στον χρόνο εκτέλεσης σε σύγκριση με τα RDBMS συστήματα ωστόσο αυτό αντισταθμίζεται από το γεγονός ότι παρουσιάζεται σημαντική αύξηση στην απόδοση συγκεκριμένων μοντέλων δεδομένων. Αυξημένη σημασία στα NoSQL συστήματα παρουσιάζει η ικανότητα να αποθηκευτούν και να ανακτηθούν μεγάλες ποσότητες δεδομένων χωρίς να δίνεται σημασία στη συσχέτιση μεταξύ αυτών οπότε και δεν υπάρχει ανάγκη για δόμηση αυτών σε μορφές όπως για παράδειγμα αυτή του πίνακα που υπάρχει στα αντίστοιχα μοντέλα των RDBMS. Είναι κατασκευασμένα με τέτοιο τρόπο ούτως ώστε να μπορούν να διαχειριστούν αρκετά μεγάλες ποσότητες δεδομένων. Οι μέθοδοι υλοποίησης και εφαρμογής τους αξιοποιούν αρχιτεκτονική η οποία διευκολύνει την κατανεμημένη λειτουργία του συστήματος. Με αυτόν τον τρόπο οι αποδόσεις του συστήματος είναι στο μέγιστο βαθμό καθώς μπορούν να υπάρξουν αναρίθμητοι servers που θα επεξεργάζονται τα δεδομένα του συστήματος.

Οι κατηγορίες στις οποίες διακρίνονται οι βάσεις δεδομένων τύπου NoSQL είναι οι εξής: **Key-value stores.** Δημιουργήθηκαν με κύρια ιδέα την ύπαρξη ενός hash table όπου υπάρχει ένα μοναδικό κλειδί και ένας δείκτης στοχεύοντας σε ένα συγκεκριμένο στοιχείο. Αυτό το είδος mapping συνήθως συνοδεύεται από μηχανισμούς cache, για την καλύτερη απόδοση του συστήματος.**Wide-column stores.** Σχεδιάστηκαν με τέτοιο τρόπο ούτως ώστε να διαχειρίζονται πολύ μεγάλα ποσά δεδομένων που είναι κατανεμημένα σε διάφορους servers. Όπως και στην κατηγορία key-value stores, έτσι κι εδώ υπάρχουν συγκεκριμένα κλειδιά (keys) που στοχεύουν όμως σε περισσότερα από ένα στοιχεία. Οι rows εδώ αναγνωρίζονται από ένα μοναδικό row key ενώ οι στήλες είναι οργανωμένες σε column families.**Document Databases.** Είναι όμοιες με τις key-value stores. Τα δεδομένα σε αυτή την περίπτωση είναι οργανωμένα από «συλλογές» key-valued «συλλογών» δεδομένων. Στην κατηγορία αυτή ανήκουν και οι MongoDb βάσεις που θα παρουσιαστούν στην συνέχεια με περισσότερη λεπτομέρεια. **Graph stores.** Είναι βασισμένη σε κόμβους (nodes), τις σχέσεις μεταξύ αυτών των κόμβων και τις ιδιότητές τους. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graph model) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα (servers – κόμβους).

3.1.3 MySQL Βάσεις Δεδομένων

Η MySQL είναι ένα από τα πιο γνωστά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων. Το SQL κομμάτι έχει την έννοια "Structured Query Language". Η SQL είναι μία γλώσσα υπολογιστών που δημιουργήθηκε προκειμένου να πραγματοποιείται η διαχείριση των δεδομένων σε ένα σύστημα σχεσιακών βάσεων δεδομένων (RDBMS). Μερικά σημαντικά πλεονεκτήματα της MySQL είναι η υψηλή απόδοση και διαθεσιμότητα, η ισχυρή προστασία δεδομένων, η ευελιξία αφού υποστηρίζει πολλές πλατφόρμες όπως είναι οι Linux και UNIX Windows και τέλος τα προγράμματα είναι ανοιχτού κώδικα.

Η MySQL υποστηρίζει εντολές τύπου SQL προκειμένου να δημιουργηθεί η βάση , χρήση της εντολής CREATE DATABASE, ενώ χρησιμοποιούνται εντολές και για τη διαχείριση αυτής όπως είναι η ALTER TABLE για την επεξεργασία της δομής ενός πίνακα. Τα δεδομένα που είανι τύπου MySQL δεν έχουν ευέλικτο σχήμα και αυτό γιατί πρέπει να δηλώνεται το σχήμα ενός πίνακα προτού γίνει εισαγωγή των δεδομένων σε αυτόν. Οι τύποι δεδομένων που υποστηρίζονται από τις MySQL βάσεις ποικίλουν και ανήκουν σε διάφορες κατηγορίες όπως ο αριθμητικός τύπος, τύπος ημερομηνίας και ώρας, αλφαριθμητικός (string) τύπος, χωρικοί τύποι (spatial types) και ο τύπος δεδομένων JSON.

Η MySQL παρέχει πολλούς τρόπους ρύθμισης του MySQL Server (mysqld), το οποίο είναι το κύριο πρόγραμμα που εκτελεί το μεγαλύτερο μέρος της εργασίας σε μια εγκατάσταση της MySQL. Ο MySQLServer περιέχει ένα σύνολο μεταβλητών συστήματος που επηρεάζουν τη λειτουργία του καθώς εκτελείται. Αυτές οι μεταβλητές μπορούν ρυθμιστούν κατά την εκκίνηση του server, και πολλές από αυτές μπορούν να αλλαχθούν κατά το χρόνο εκτέλεσης για δυναμική αναδιαμόρφωση του server. Επίσης, ο MySQL Server έχει και ένα σύνολο μεταβλητών κατάστασης που παρέχουν πληροφορίες σχετικά με τη λειτουργία του. Οι τρόποι που μπορούμε να ρυθμίσουμε τις μεταβλητές συστήματος και

κατάστασης είναι μέσω αρχείων ρυθμίσεων ή μέσω εντολών σε command line ή και μέσω γραφικών διεπαφών όπως είναι το MySQL Workbench.

3.1.4 MongoDB Βάσεις Δεδομένων

Η MongoDB είναι μία ανοιχτού κώδικα NoSQL βάση δεδομένων και ανήκει στην κατηγορία των document βάσεων δεδομένων όπως αναφέρθηκε προηγουμένος. Το αντίστοιχο της εγγραφής (record) των σχεσιακών βάσεων εδώ ονομάζεται έγγραφο (document) και περιέχει πεδία που απαρτίζονται από ζεύγη κλειδιών-τιμών (key-value pairs) ενώ το αντίστοιχο του πίνακα (table) των σχεσιακών βάσεων, εδώ ορίζεται ως συλλογή (collection). Τα έγγραφα της MongoDB είναι παρόμοια με τα αντικείμενα της μορφής JSON και πιο συγκεκριμένα αποθηκεύονται στη μορφή Binary-Encoded JSON (BSON). Οι τιμές των πεδίων μπορούν να περιέχουν άλλα έγγραφα, πίνακες και πίνακες από έγγραφα.

Τα κύρια πλεονεκτήματα της χρήσης εγγράφων είναι ότι αντιστοιχούν στους (έμφυτους - native) τύπους δεδομένων πολλών μοντέρνων γλωσσών προγραμματισμού (όπως η ιδιαίτερα δημοφιλής τελευταία JavaScript) διευκολύνοντας πολύ τον προγραμματισμό εφαρμογών. Επίσης η δυνατότητα για ενσωματωμένα έγγραφα και πίνακες μειώνει την ανάγκη για δαπανηρές πράξεις τύπου join. Τέλος η δυνατότητα χρήσης αδόμητου ή ημιδομημένου σχήματος υποστηρίζει την εύκολη επέκταση των εφαρμογών.

Η MongoDB παρέχει υψηλή απόδοση, υψηλή διαθεσιμότητα και αυτόματη κλιμάκωση και μερικά από τα χαρακτηριστικά της βάσης που τα συντελούν είναι η υποστήριξη ενσωμάτωσης αντικειμένων στο μοντέλο δεδομένων της (nested documents), δυνατότητα η οποία μειώνει την ανάγκη για πολλαπλές αναγνώσεις/εγγραφές στους χώρους αποθήκευσης. Η παροχή δεικτών (indexes) οι οποίοι μπορούν να δεικτοδοτήσουν κλειδιά και σε ενσωματωμένα έγγραφα (documents).Η υπηρεσία λειτουργίας αντιγράφων της βάσης (replication) παρέχει αυτόματη ανάκαμψη από βλάβες (automatic failover) και πλεονασμό δεδομένων (data redundancy). Η παροχή οριζόντιας κλιμάκωσης ως βασική της υπηρεσία και παρέχει την δυνατότητα κατακερματισμού των δεδομένων (sharding) σε ένα σύνολο (cluster) υπολογιστών.

Η MongoDB αποθηκεύει τα δεδομένα σε έγγραφα χρησιμοποιώντας την αναπαράσταση BSON στα δεδομένα και ομαδοποιεί τα έγγραφα μέσα σε συλλογές.. Ως έγγραφα (Documents) ορίζονται τα εξής: Η MongoDB αποθηκεύει τα δεδομένα σε έγγραφα τα οποία απαρτίζονται από ζεύγη κλειδιών-τιμών (key-value pairs). Τα έγγραφα μπορούν να φτάσουν σε μέγεθος μέχρι τα 16MB και το καθένα θα πρέπει να έχει ένα _id πεδίο το οποίο δεικτοδοτείται (indexed) αυτόματα. Το πεδίο _id παράγεται αυτόματα αν δεν έχει προσδιοριστεί και χρησιμοποιείται ως το μοναδικό αναγνωριστικό για το έγγραφο. Τα δεδομένα σε ένα έγγραφο αποθηκεύονται σε μια αναπαράσταση παρόμοιας της JavaScript Object Notation (JSON) που ονομάζεται BSON. Ως συλλογές (Collections): Χρησιμοποιούνται για να ομαδοποιούν πολλά τα έγγραφα και είναι παρόμοια με τους πίνακες (tables) των RDBMS. Οι συλλογές μπορούν να περιέχουν έγγραφα με διαφορετική δομή ως προς το σχήμα (schema), ωστόσο είναι καλή πρακτική να αποθηκεύονται έγγραφα με την ίδια δομή. Ως BSON: Η MongoDB χρησιμοποιεί τη μορφή BSON για να αποθηκεύει τα έγγραφα. Η BSON είναι μια δυαδικά κωδικοποιημένη μορφή που

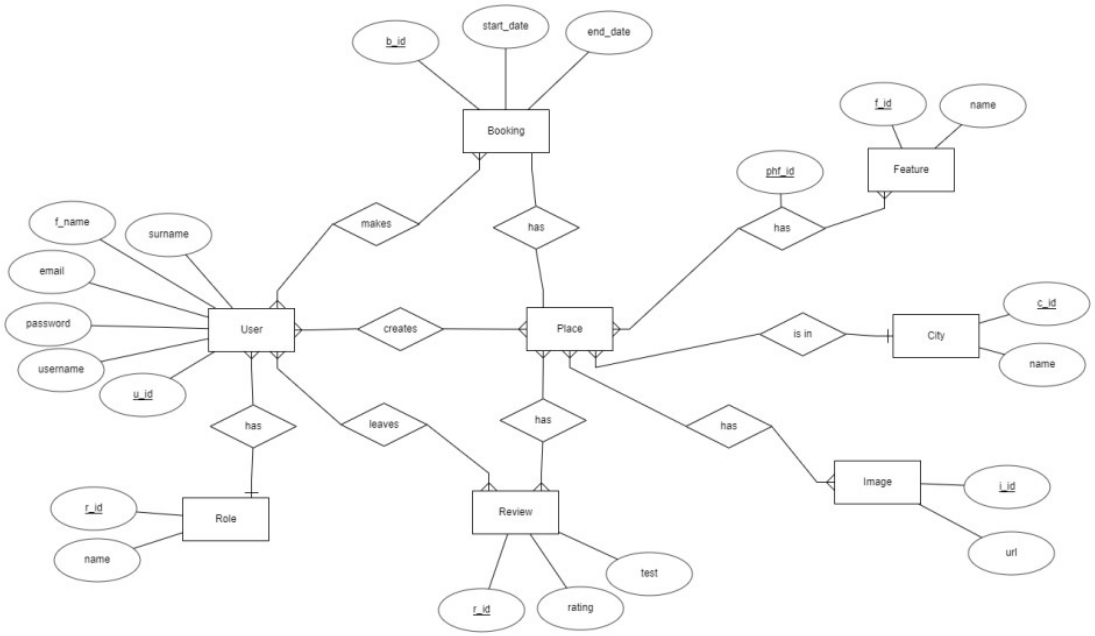
επεκτείνει το γνωστό JSON.

Όπως αναφέρθηκε προηγουμένως τα δεδομένα στην MongoDB έχουν ένα ευέλικτο σχήμα (schema). Σε αντίθεση με τις SQL σχεσιακές βάσεις δεδομένων, όπου πρέπει να καθορίζουμε και να δηλώνουμε το σχήμα ενός πίνακα προτού εισάγουμε τα δεδομένα, οι συλλογές της MongoDB δεν προκαθορίζουν την δομή του εγγράφου. Αυτή η ευελιξία διευκολύνει τη χαρτογράφηση των εγγράφων σε μια οντότητα ή ένα αντικείμενο. Κάθε έγγραφο μπορεί να ταιριάξει τα πεδία δεδομένων της εκπροσωπούμενης οντότητας, ακόμη και αν τα δεδομένα είναι διαφορετικά. Στην πράξη, όμως, τα έγγραφα σε μια συλλογή μοιράζονται μια παρόμοια δομή. Η βασική απόφαση για το σχεδιασμό μοντέλο δεδομένων για MongoDB εφαρμογές περιστρέφεται γύρω από τη δομή των εγγράφων και πως η εφαρμογή παριστάνει τις σχέσεις μεταξύ των δεδομένων. Οι δύο τεχνικές που επιτρέπουν στις εφαρμογές να παριστάνουν αυτές τις σχέσεις είναι τα έγγραφα με αναφορές (references documents) και τα ενσωματωμένα έγγραφα (embedded documents). Οι αναφορές ‘αποθηκεύουν’ τις σχέσεις των δεδομένων με το να περιλαμβάνουν συνδέσμους (links) ή αναφορές (references) από το ένα έγγραφο στο άλλο. Οι εφαρμογές μπορούν να αξιοποιήσουν αυτές τις αναφορές για να έχουν πρόσβαση σε δεδομένα που αιτούνται. Γενικά, αυτά ονομάζονται κανονικοποιημένα μοντέλα δεδομένων (normalized data models).

3.2 Μοντελοποίηση

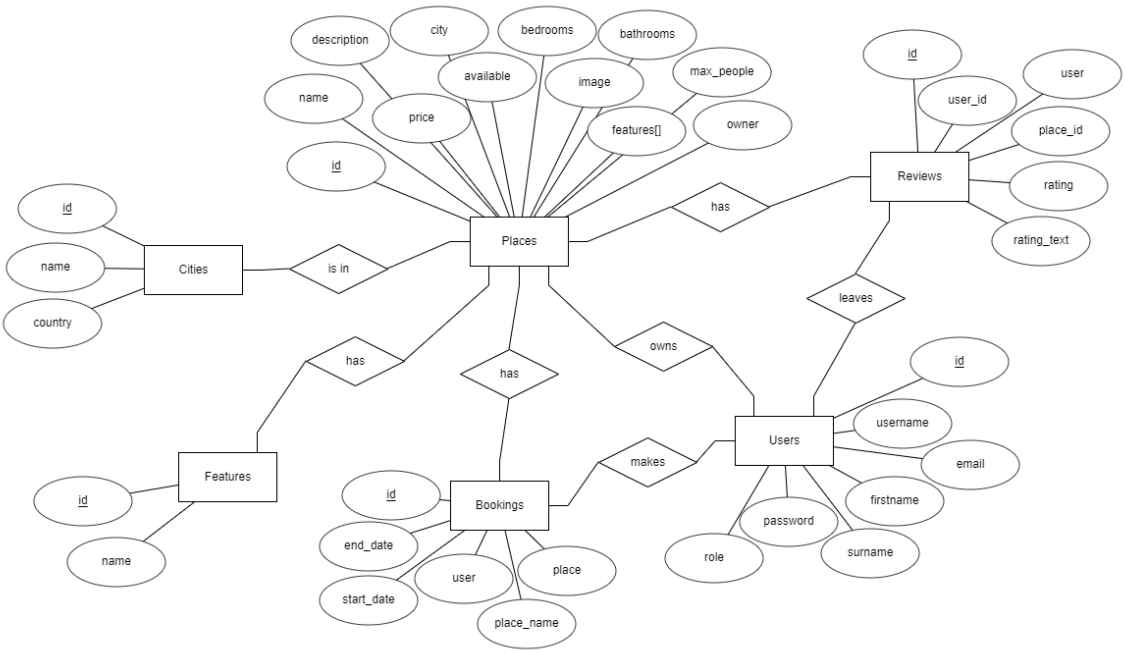
Το μοντέλο οντοτήτων-συσχετίσεων (μοντέλο Ο/Σ – ER model) είναι ένα αφαιρετικό ιδεατό μοντέλο δεδομένων, τα οποία έχουν καθορισμένη δομή. Στη μηχανική λογισμικού χρησιμοποιείται για να παρέχει ένα εννοιολογικό σχήμα κατά τη σχεδίαση βάσεων δεδομένων, ως μοντέλο δεδομένων ενός συστήματος και των απαιτήσεών του με top-down προσέγγιση. Ένα διάγραμμα που δημιουργείται με αυτή τη διαδικασία σχεδίασης καλείται διάγραμμα οντοτήτων-συσχετίσεων, ή διάγραμμα Ο/Σ ή ΟΣΔ εν συντομία χρησιμοποιείται στο πρώτο στάδιο σχεδίασης ενός συστήματος πληροφοριών, κατά την ανάλυση των απαιτήσεών του. Σκοπός του είναι να περιγράφει τις αναγκαίες πληροφορίες οι οποίες πρόκειται να αποθηκευτούν στη βάση δεδομένων ή τον τύπο τους. Η μοντελοποίηση δεδομένων γίνεται για την περιγραφή των χρησιμοποιούμενων όρων και των σχέσεών τους σε έναν ορισμένο τομέα ενδιαφέροντος. Στην περίπτωση σχεδιασμού ενός συστήματος πληροφοριών, που στηρίζεται σε μια βάση δεδομένων, το εννοιολογικό μοντέλο δεδομένων χαρτογραφείται σε προχωρημένο στάδιο σε ένα λογικό μοντέλο δεδομένων, όπως το σχεσιακό μοντέλο δεδομένων.

3.2.1 Διάγραμμα ΜΟΣ (PHP)



Εικόνα 1: Διάγραμμα ΜΟΣ για την εφαρμογή PHP

3.2.2 Διάγραμμα ΜΟΣ (Node.js)



Εικόνα 2: Διάγραμμα ΜΟΣ για την εφαρμογή Node.js

3.3 Κανονικοποίηση

Στις εμπορικές εφαρμογές σχεσιακών βάσεων δεδομένων κατασκευάζουμε την Τρίτη Κανονική Μορφή. Οι πίνακες μας στη μορφή αυτή δεν έχουν άχρηστους πλεονασμούς και επομένως η Τρίτη Κανονική Μορφή είναι αρκετά οικονομική σε αποθήκευση δεδομένων. Επιπλέον, η μορφή αυτή μας εξασφαλίζει ότι μπορούμε εύκολα και γρήγορα να αναζητούμε στοιχεία. Στη συνέχεια παραθέτουμε τους πιο σημαντικούς πίνακες της Τρίτης Κανονικής Μορφής που σχεδιάσαμε για το σύστημά μας. Επιπλέον, παραθέτουμε παραδείγματα με δείγμα δεδομένων ανά πίνακα του συστήματος μας.

3.4 CMS

Το CMS (Content Management System) είναι διαδυκτιακή εφαρμογή που επιτρέπει στον διαχειριστή της να τροποποιεί τον ιστότοπό του (website), να μπορεί δηλαδή να μεταποιήσει το γενικό περιεχόμενο του site του είτε αυτό είναι κάποιο κείμενο είτε κάποιο πολυμέσο (π.χ. βίντεο ή εικόνα). Με άλλα λόγια είναι ένα εργαλείο που βοηθάει στη δημιουργία ενός website χωρίς όμως να χρειαστεί κάποιος να στήσει προγραμματιστικά τον ιστότοπό του από το μηδέν.

3.5 MVC

Για να καταλάβει κάποιος το μοντέλο MVC πρέπει πρώτα να γνωρίσει τις έννοιες των Views, των Controllers και των Models. Τα views είναι οι ιστοσελίδες που θα δει ο χρήστης για να μπορέσει να έρθει σε επαφή με το λειτουργικό κομμάτι της εφαρμογής. Οι controllers σε μία web εφαρμογή είναι ο συνδετικός κρίκος μεταξύ των models και των views, είναι στην ουσία οι διαχειριστές όλου του interface. Τα models αποτελούνται από όλες τις κλάσεις που υπάρχουν και χρησιμοποιούνται από την εφαρμογή για να σταλεί πληροφορία μεταξύ views. Συνήθως αυτή η πληροφορία προέρχεται από τη βάση δεδομένων της εφαρμογής.

Το MVC χωρίζει τη βασική λειτουργικότητα της εφαρμογής σε τρία κομμάτια, αυτό του Model, του View και του Controller. Αυτά τα τρία κομμάτια συνεργάζονται μεταξύ τους για να πάρουν το input από τον χρήστη και να παράγουν κάποιο response σε αυτόν. Πιο συγκεκριμένα ο τρόπος λειτουργίας ενός συστήματος βασισμένου σε σύστημα MVC λειτουργεί ως εξής: ο χρήστης μέσω του View βλέπει το interface της πλατφόρμας. Όταν μέσω μίας διαδικασίας ο χρήστης δώσει εντολή στο σύστημα για την ενέργεια που θέλει να πραγματοποιήσει η πληροφορία του τι είναι αυτό που ο χρήστης θέλει να κάνει, μεταφέρεται στους Controllers που είναι συνδεδεμένοι με το αντίστοιχο interface. Αυτοί με τη σειρά τους επεξεργάζονται τη συγκεκριμένη πληροφορία καθώς και τα Models (βάση δεδομένων) που υπάρχουν στην εφαρμογή. Αφού συμβεί αυτό η διαδικασία ολοκληρώνεται με την ενημέρωση του νέου view από τα models πάλι μέσω των αντίστοιχων controllers που θα στείλουν τις πληροφορίες που χρειάζονται.

4 Ανάλυση Απαιτήσεων Συστήματος

Η διπλωματική εργασία έχει ως αντικείμενο όπως αναφέρθηκε τη σύγκριση των βάσεων MySQL και MongoDB με την πραγματοποίηση τριών διαφορετικών εφαρμογών. Οι τρεις αυτές εφαρμογές θα μελετηθούν σε αυτό το κεφάλαιο ως προς την αρχιτεκτονική τους. Αυτή είναι τα υποσυστήματα από τα οποία αποτελούνται τα οποία θα περιγραφούν εν μέρη εδώ και θα αναλυθούν σε επόμενη ενότητα.

4.1 Αρχιτεκτονική

Η βασική ιδέα της υλοποίησης ήταν μέσω μίας εφαρμογής κοινού περιεχομένου να αναδειχθούν οι λειτουργίες διαφορετικών βάσεων δεδομένων και να γίνει σύγκριση αυτών. Με τη λογική αυτή η αρχιτεκτονική που παρουσιάζει η κάθε εφαρμογή ως προς το front end κομμάτι, δηλαδή το κομμάτι που βλέπει ο χρήστης και έρχεται σε επαφή με αυτό είναι κοινή και για τις τρεις. Ωστόσο καθώς όλες μας οι εφαρμογές υποστηρίζουν δύο διαφορετικές μορφής χρήστες θα πρέπει η κάθε αρχιτεκτονική να παρουσιαστεί και για τους δύο. Αυτοί οι χρήστες είναι οι απλοί users της εφαρμογής, δηλαδή οι χρήστες που μπαίνουν για να χρησιμοποιήσουν τις λειτουργίες της και οι admins, αυτοί δηλαδή που διαχειρίζονται την λειτουργία της εφαρμογής.

Οι αρχιτεκτονικές αυτές είναι οι εξής:

1. Register
2. Login – ταυτοποίηση
3. Create – δημιουργία περιεχομένου
4. Update – επεξεργασία περιεχομένου
5. Delete – διαγραφή περιεχομένου
6. Search
7. Booking
8. Review

4.2.1 Περιγραφή Λειτουργιών (Users)

Εδώ θα γίνει η περιγραφή των παραπάνω αρχιτεκτονικών.

4.2.1.1 Register

Εδώ ο user καλείται να δώσει τα προσωπικά του στοιχεία του, όνομα, email, κωδικό και άλλα, ούτως ώστε να αποκτήσει πρόσβαση στην εφαρμογή και να μπορεί να πραγματοποιήσει τις λειτουργίες της.

4.2.1.2 Login

Εδώ εφόσον ο user έχει ήδη από προηγούμενη φορά δημιουργήσει “λογαριασμό” στην εφαρμογή, έχει δηλαδή καταχωρίσει τα προσωπικά του στοιχεία μπορεί να τα καταχωρήσει και πάλι και έτσι αν αποκτήσει πρόσβαση στο σύστημα.

4.2.1.3 Create

Το υποσύστημα create δίνει τη δυνατότητα στο χρήστη να δημιουργήσει ένα αντικείμενο, στην προκειμένη περίπτωση ένα οίκημα που θέλει. Ο user καλείται να συμπληρώσει μία φόρμα με τα στοιχεία του οικήματος, όνομα, περιγραφή, πόλη κ.α., και έτσι δημιουργεί το στοιχείο place.

4.2.1.4 Update

Το update δομικά είναι παρόμοιο με το create καθώς ο user καλείται να συμπληρώσει την ίδια φόρμα με τα στοιχεία του οικήματος με τη διαφορά ότι εδώ πρέπει να είναι και κάτοχος του οικήματος στο οποίο θέλει να αλλάξει κάποιες από τις πληροφορίες που έχει ήδη υποβάλλει στο σύστημα.

4.2.1.5 Delete

Εδώ ο user έχει την επιλογή με το πάτημα ενός κουμπιού που βλέπει στην εφαρμογή να διαγράψει ένα δεδομένο που έχει καταχωρίσει στο σύστημα.

4.2.1.6 Search

Το search είναι ένα σύστημα το οποίο δίνει τη δυνατότητα στον χρήστη να ψάξει μία πληροφορία που αναζητεί. Αυτό γίνεται πληκτρολογώντας την πληροφορία που ζητάει στο πεδίο search.

4.2.1.7 Booking

Το σύστημα αποτελείται από ένα παράθυρο με δύο ημερολόγια στο ένα βλέπουμε ημερομηνίες άφιξης και στο άλλο ημερομηνίες αποχώρησης τις οποίες επιλέγει ο χρήστης.

4.2.1.8 Review

Τα reviews είναι ένα σύστημα που δημιουργήθηκε ούτως ώστε να μπορούν να δουν όλοι οι χρήστες της εφαρμογής την άποψη ή και τις απόψεις των άλλων χρηστών που έχουν ήδη επισκεφθεί το οίκημα στο οποίο ανήκει.

4.2.2 Περιγραφή Λειτουργιών (Admins)

4.2.2.1 Register

Δεν υπάρχει συγκεκριμένη αρχιτεκτονική για την είσοδο του admin στο σύστημα τα στοιχεία του για να έχει πρόσβαση στην εφαρμογή δηλώνονται με διαφορετικό τρόπο στο σύστημα όπου θα αναλυθεί σε επόμενο στάδιο.

4.2.2.2 Login

Οι Admins βλέπουν το ίδιο παράθυρο σύνδεσης με τους υπόλοιπους χρήστες.

4.2.2.3 Create

Η βασική διαφορά του υποσυστήματος create για τους Admins είναι ότι έχουν τη δυνατότητα να εισάγουν στοιχεία, όπως οι πόλεις και features.

4.2.2.4 Update

Στα ίδια πλαίσια με το create καθώς οι Admins διαχειρίζονται ολόκληρη την εφαρμογή μπορούν να κάνουν update και στις άλλες κατηγορίες πέραν των οικημάτων που οι ίδιοι διαθέτουν αλλά να ενημερώσουν το σύστημα και για κάποιο οίκημα που δεν διαθέτουν οι ίδιοι.

4.2.2.5 Delete

Οι Admins του συστήματος μπορούν να διαγράψουν οποιαδήποτε πληροφορία του συστήματος με το πάτημα ενός κουμπιού.

4.2.2.6 Search

Το search είναι ένα σύστημα το οποίο δίνει τη δυνατότητα στον χρήστη να ψάξει μία πληροφορία που αναζητεί. Αυτό γίνεται πληκτρολογώντας την πληροφορία που ζητάει στο πεδίο search.

4.2.2.7 Booking

Το σύστημα αποτελείται από ένα παράθυρο με δύο ημερολόγια στο ένα βλέπουμε ημερομηνίες άφιξης και στο άλλο ημερομηνίες αποχώρησης τις οποίες επιλέγει ο χρήστης.

4.2.2.8 Review

Τα reviews όπως αναφέρθηκε είναι ένα σύστημα που δημιουργήθηκε ούτως ώστε να μπορούν να δουν όλοι οι χρήστες της εφαρμογής την άποψη ή και τις απόψεις των άλλων χρηστών που έχουν ήδη επισκεφθεί το οίκημα στο οποίο ανήκει.

5 Σχεδίαση Συστήματος

Εδώ θα παρουσιαστούν οι τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή της κάθε εφαρμογής.

Χρήσιμες έννοιες κεφαλαίου:

Front end: είναι όρος που χρησιμοποιείται από προγραμματιστές για να περιγράψουν το κομμάτι της εφαρμογής στο οποίο υπάρχει όλο το software ή hardware που έχει να κάνει με τον χρήστη. Οτιδήποτε βλέπει ο χρήστης βρίσκεται σε αυτό το κομμάτι.

Back end: αναφέρεται στα κομμάτια της εφαρμογής που της επιτρέπουν να λειτουργεί και στα οποία δεν έχει πρόσβαση ο χρήστης. Τα περισσότερα δεδομένα, ο κώδικας καθώς και τα συστήματα εκτέλεσης βρίσκονται σε αυτό το κομμάτι. Επίσης επικοινωνεί με το front end κομμάτι αφού είναι υπεύθυνο για τις λειτουργίες του.

5.1 Γλώσσες

Παρουσιάζονται όλες οι γλώσσες που χρησιμοποιήθηκαν για την υλοποίηση κάθε μίας εκ των τριών εφαρμογών.

5.1.1 HTML

Η HTML είναι η πιο κοινά διαδεδομένη γλώσσα για την δημιουργία web σελίδων. Τα αρχικά της σημαίνουν Hyper Text Markup Language. Η γλώσσα περιγράφει τη δομή μίας web σελίδας. Συντίθεται από διάφορα στοιχεία τα οποία καθοδηγούν τον browser ούτως ώστε να δείξει το περιεχόμενο της σελίδας. Η δομή της HTML είναι η εξής:

```
1  <html>
2  <head>
3      <!-- Information about the page -->
4      <title>HTML Page</title>
5  </head>
6  <body>
7      This is the content of the page.
8  </body>
9  </html>
```

Εικόνα 3: Παράδειγμα από κώδικα HTML

5.1.2 CSS

Η CSS είναι μία γλώσσα που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός προγράμματος που έχει γραφτεί με HTML. Είναι δηλαδή η γλώσσα που χρησιμοποιείται για να μορφοποιήσει και να δώσει δομή σε μία ιστοσελίδα. Η γλώσσα αυτή προσφέρει έναν αρκετά εύκολο τρόπο ούτως ώστε η σελίδα να πάρει την ιδανική μορφή και να προσελκύσει το ενδιαφέρον του χρήστη.

5.1.3 PHP

Η PHP είναι μία ευρέως διαδεδομένη γλώσσα προγραμματισμού που χρησιμοποιείται κυρίως για την κατασκευή web σελίδων. Τις περισσότερες φορές χρησιμοποιείται μαζί με κώδικα HTML. Στην ουσία η PHP δίνει λειτουργικότητα στα αντικείμενα που δημιουργούνται με κώδικα HTML. Η PHP είναι μία πολύ απλή γλώσσα που μπορεί να χρησιμοποιηθεί εύκολα από προγραμματιστές που δεν έχουν μεγάλη εμπειρία ωστόσο εάν χρησιμοποιηθεί από πιο εξειδικευμένους με γλώσσες προγραμματιστές προσφέρει προχωρημένες δυνατότητες.

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>Example</title>
5       </head>
6       <body>
7
8           <?php
9           echo "Hi, I'm a PHP script!";
10          ?>
11
12      </body>
13  </html>
```

Εικόνα 4: Παράδειγμα από κώδικα PHP

Αυτό είναι ένα παράδειγμα κώδικα PHP μέσα σε κώδικα HTML. Η HTML φτιάχνει μία σελίδα με τίτλο Example και η PHP θα τυπώσει εκέι το μήνυμα “Hi, I’m a PHP script!”.

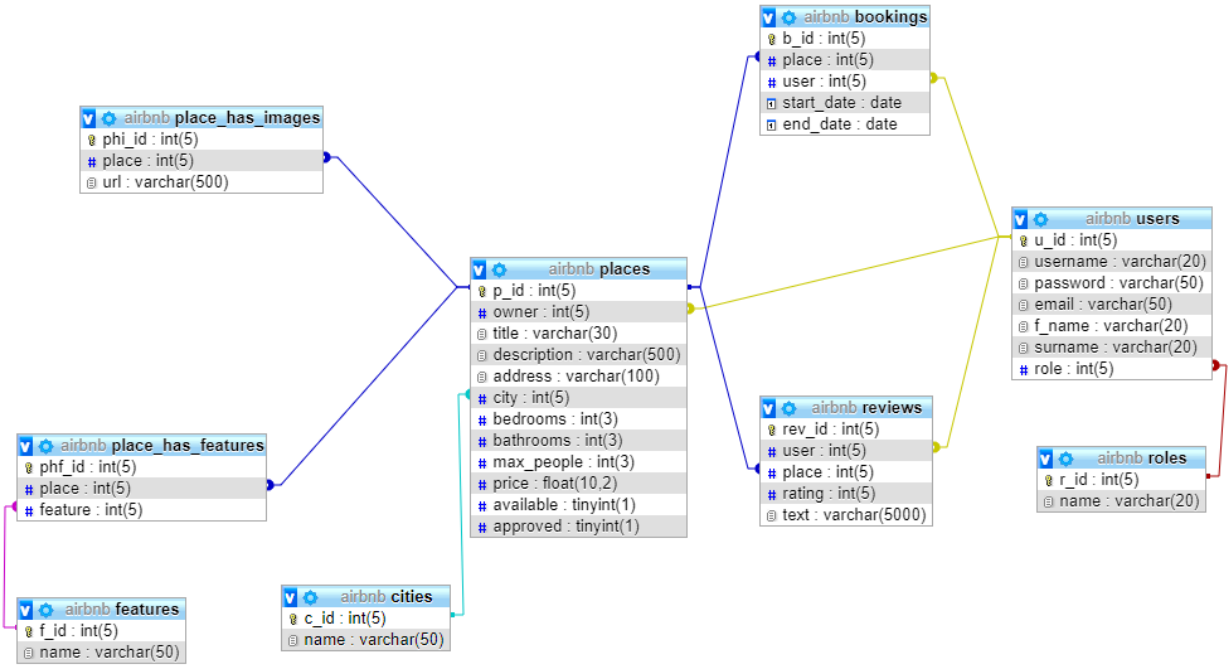
5.1.4 JavaScript

Η JavaScript είναι μία γλώσσα που χρησιμοποιείται για να προσφέρει διαδραστικότητα στις web σελίδες. Χρησιμοποιείται τόσο στην πλευρά του client όσο στην πλευρά του server.

5.2 Βάσεις

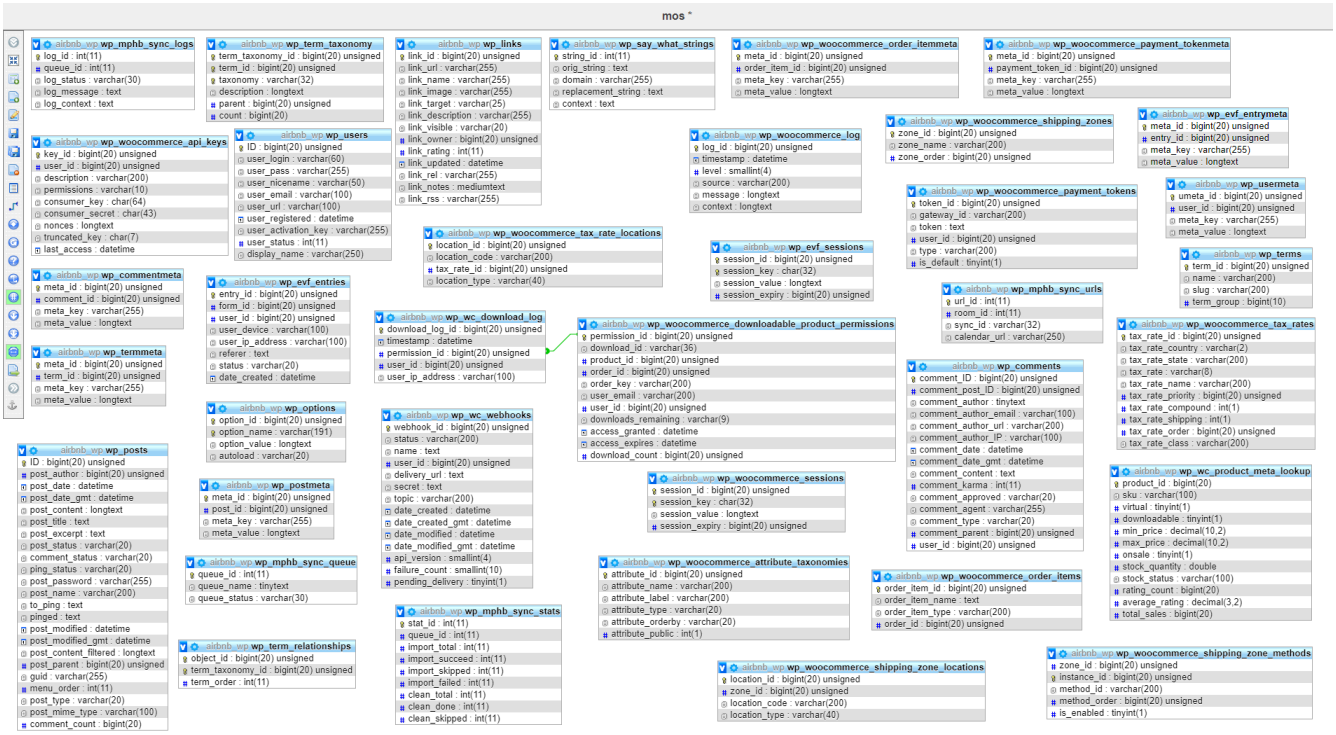
Όπως έχει ήδη αναφερθεί οι βάσεις δεδομένων που χρησιμοποιήθηκαν είναι τύπου MySQL για το PHP πρόγραμμα και MongoDB για το Node.js πρόγραμμα. Το wordpress γεννάει τη δική του βάση δεδομένων τύπου MySQL. Η δομή των βάσεων για τις εφαρμογές php και wordpress εμφανίζεται παρακάτω.

5.2.1 MySQL - PHP



Εικόνα 5: Δομή της βάσης MySQL της εφαρμογής PHP

5.2.2 MySQL - Wordpress



Εικόνα 6: Δομή της βάσης MySQL για την εφαρμογή Wordpress

5.3 Μεθοδολογίες – Τεχνικές

Η πιο σημαντική από τις μεθοδολογίες – τεχνικές που χρησιμοποιήθηκαν είναι αυτή των sessions. Τα sessions αποθηκεύουν, προσωρινά, πληροφορίες σε μεταβλητές τύπου sessions συνήθως κυρίως για να χρησιμοποιηθούν δεδομένα σε διάφορες web σελίδες ενός προγράμματος. Για παράδειγμα websites που απαιτούν για την είσοδο του χρήστη να εισάγει όνομα χρήστη και κωδικό (διαδικασία login) χρησιμοποιούν μεταβλητές session για να μεταφερθούν οι πληροφορίες αυτές του χρήστη από σελίδα σε σελίδα αλλά μόνο ενώ ο χρήστης παραμένει συνδεδεμένος στην εφαρμογή καθώς κατά την έξοδό του απελευθερώνεται οποιαδήποτε πληροφορία με βάση το συγκεκριμένο χρήστη είχε αποθηκευτεί προσωρινά σε session εφαρμογή.

5.4 Εργαλεία

Για την συγγραφή των προγραμμάτων αλλά και για την προσομοίωση τους ούτως ώστε να γίνουν τεστ στην λειτουργικότητα και την εμφάνιση χρησιμοποιήθηκαν κάποια βοηθητικά εργαλεία και προγράμματα. Αυτά παρουσιάζονται στην συνέχεια.

5.4.1 XAMPP

Το XAMPP είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού το οποίο περιέχει το εξυπηρετητή ιστοσελίδων HTTP Apache, την βάση δεδομένων MySQL και ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP και Perl. Προσφέρει τη δυνατότητα δοκιμής των ιστοσελίδων που δημιουργήθηκαν, τοπικά στον υπολογιστή χωρίς αυτές να είναι προσβάσιμες από κάποιον υπολογιστή εκτός του δικτύου. Ταυτόχρονα παρέχει δυνατότητα φιλοξενίας μίας βάσεως δεδομένων SQL.

5.4.2 phpMyAdmin

Το phpMyAdmin είναι ένα περιβάλλον με οποίο μπορούμε να διαχειριστούμε εύκολα τις βάσεις τύπου MySQL σε έναν database server. Το phpMyAdmin υπάρχει ενσωματωμένο στο XAMPP. Είναι στην ουσία μία εφαρμογή που διευκολύνει την επεξεργασία της βάσης δεδομένων και βοηθάει τον χρήστη να τσεκάρει δεδομένα, να ελέγχει για λάθη και να βλέπει εάν πραγματοποιούνται οι αλλαγές σε αυτή που θα έπρεπε.

5.4.3 Visual Studio Code

Το Visual Studio Code είναι μία πλατφόρμα συγγραφής κώδικα που παρέχει στον χρήστη λειτουργίες όπως αυτή του auto complete, compiling, ανίχνευση λαθών πάνω στον κώδικα βοηθώντας έτσι τον προγραμματιστεί να γράφει πιο γρήγορα ενώ ταυτόχρονα και να ελέγχει για τυχόν συντακτικά λάθη που μπορεί να του έχουν ξεφύγει κατά τη διάρκεια της συγγραφής.

5.4.4 MongoDB Compass

Το MongoDB Compass είναι η πλατφόρμα που χρησιμοποιήθηκε για τον έλεγχο και την διαχείριση της βάσης MongoDB που κατασκευάστηκε. Ο χρήστης μπορεί να κάνει έλεγχο στη βάση, να δει τις εγγραφές, να ελέγξει με queries τα δεδομένα και έτσι να είναι σίγουρος ότι το πρόγραμμά του λειτουργεί σωστά.

6 Υλοποίηση

Στο κεφάλαιο αυτό θα γίνει ανάλυση της υλοποίησης των εφαρμογών τύπου Airbnb που κατασκευάστηκαν. Οι εφαρμογές αυτές είναι:

1. Εφαρμογή PHP – MySQL
2. Εφαρμογή Wordpress
3. Εφαρμογή Node.js – MongoDB

6.1 PHP – MySQL

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε ένας υπολογιστής με λειτουργικό σύστημα Windows 10. Σε αυτόν τον υπολογιστή εγκαταστάθηκε το XAMPP που είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού, λογισμικού ανοικτού κώδικα και ανεξαρτήτου πλατφόρμας το οποίο περιέχει τον εξυπηρετητή ιστοσελίδων HTTP Apache, την βάση δεδομένων MySQL και ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP. Ουσιαστικά χρησιμοποιήθηκε για να μπορεί να γίνει testing των ιστοσελίδων που δημιουργήθηκαν χωρίς να χρειαστεί να γίνει αγορά ενός server και domain για την φιλοξενία της εφαρμογής.

Για τη συγγραφή του php κώδικα καθώς και του html χρησιμοποιήθηκαν περιβάλλοντα συγγραφής κώδικα όπως το Visual Studio Code, που παρέχουν δυνατότητες όπως αυτόματη μορφοποίηση του κώδικα, autocomplete και δυνατότητες αναγνώρισης λαθών στον κώδικα.

Δημιουργία και Στήσιμο Βάσης

Η βάση που επιλέχθηκε για τον συγκεκριμένο τύπο εφαρμογής είναι τύπου MySQL. Η δομή είναι ως εξής:

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	bookings	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8_general_ci	48 KiB	-
☐	cities	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	16 KiB	-
☐	features	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	16 KiB	-
☐	places	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8_general_ci	48 KiB	-
☐	place_has_features	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8_general_ci	48 KiB	-
☐	place_has_images	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8_general_ci	32 KiB	-
☐	reviews	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8_general_ci	48 KiB	-
☐	roles	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_general_ci	16 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	32 KiB	-
	9 tables	Sum	46	InnoDB	utf8_general_ci	304 KiB	0 B

Εικόνα 7: Λίστα πινάκων της βάσης της εφαρμογής PHP

Αποτελείται όπως βλέπουμε και στην εικόνα από τους πίνακες:

1. bookings
2. cities
3. features
4. places
5. place_has_features
6. place_has_images
7. reviews
8. roles
9. users

Η δομή και η συνδέσεις των πινάκων θα αναλυθούν αμέσως τώρα. Για κάθε πίνακα θα δίνονται πρώτα οι κατάλληλες διευκρινήσεις και στη συνέχεια μία εικόνα στην οποία φαίνεται η δομή του του καθενός.

Ο πίνακας **bookings** περιέχει τις πληροφορίες που είναι απαραίτητες για να πραγματοποιηθεί η ενέργεια "booking" να μπορέσει δηλαδή ο χρήστης να υλοποιήσει την κράτηση του οικήματος που τον ενδιαφέρει. Περιέχει τα πεδία b_id, place, user, start_date, end_date. Το πεδίο b_id είναι και το κύριο κλειδί της βάσης, το πεδίο δηλαδή που είναι μοναδικό για την κάθε εγγραφή του πίνακα. Στο πεδίο place αποθηκεύεται η πληροφορία που μας δείχνει για ποιο μέρος/οίκημα πραγματοποιούμαι τη συγκεκριμένη κράτηση ενώ στο πεδίο user αποθηκεύεται η πληροφορία για το ποιος χρήστης της εφαρμογής πραγματοποιεί εκείνη τη στιγμή την συγκεκριμένη κράτηση. Τα πεδία start_date και end_date, όπως δηλώνει και το όνομα τους αποθηκεύουν την πληροφορία για την μέρα άφιξης και τη μέρα αποχώρησης του ενδιαφερόμενου.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	b_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	place	int(5)			No	None			Change Drop More
☐	3	user	int(5)			No	None			Change Drop More
☐	4	start_date	date			No	None			Change Drop More
☐	5	end_date	date			No	None			Change Drop More

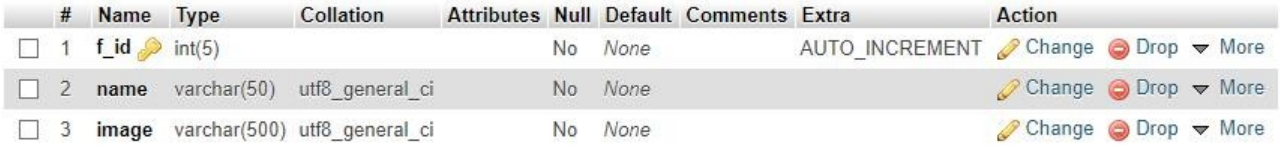
Εικόνα 8: Πεδία του πίνακα bookings

Στον πίνακα **cities** έχουμε μόνο δύο πεδία, το κύριο κλειδί που είναι απλά ένας ξεχωριστός κωδικός για κάθε πόλη καθώς και το πεδίο που περιέχει το όνομα της πόλης. Ο πίνακας αυτός δημιουργήθηκε ούτως ώστε να φιλοξενούνται οι πληροφορίες σχετικά με το ποιες πόλεις μπορεί να βρει κάποιος στην εφαρμογή.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	c_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(50)	utf8_general_ci		No	None			Change Drop More

Εικόνα 9: Πεδία του πίνακα cities

Τα **features** είναι ένας πίνακας που περιέχει τις πληροφορίες για έξτρα παροχές – ανέσεις που μπορεί να βρει κάποιος σε ένα οίκημα. Θα μπορούσε να αποτελεί πεδίο του πίνακα places ωστόσο επειδή η σχέση place και feature είναι τύπου NpN είναι αναγκαία η δημιουργία ξεχωριστού πίνακα. Η σχέση NpN δηλώνει ότι ένα feture μπορεί να ανήκει σε πολλά places αλλά και ένα place μπορεί να έχει πολλά features οπότε και είναι αδύνατο σε ένα πεδίο να αποθηκευτούν πολλαπλές διαφορετικές πληροφορίες. Τα πεδία του πίνακα feature είναι το f_id που αυξάνεται αυτόματα με κάθε εγγραφή και είναι μοναδικό για την κάθε εγγραφή, το πεδίο name που είναι η περιγραφή του κάθε feature και το πεδίο image που περιέχει το url της εικόνας που δείχνει το αντίστοιχο feature.



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	f_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	3	image	varchar(500)	utf8_general_ci		No	None			Change Drop More

Εικόνα 10: Πεδία του πίνακα features

Ο πίνακας **places** περιέχει τα πεδία p_id, owner, title, description, address, city, bedrooms, bathrooms, max_people, price, available, approved. Το πρώτο πεδίο, p_id είναι ο ξεχωριστός κωδικός του κάθε μέρους, στο δεύτερο πεδίο έχουμε το όνομα του ιδιοκτήτη, στο τρίτο την ονομασία του μέρους, δηλαδή την ονομασία με την οποία θέλουμε να παρουσιάζεται το μέρος αυτό, στο πεδίο description ο χρήστης μπορεί να εισάγει μία μίνι περιγραφή του μέρους του. Το πεδίο address έχει τη διεύθυνση ενώ το πεδίο city την πόλη του. Τα πεδία bedrooms, bathrooms, max_people και price έχουν την πληροφορία για τον αριθμό των δωματίων, μπάνιων, μέγιστο αριθμό ατόμων που φιλοξενεί και την τιμή του μέρους αντίστοιχα. Τέλος στο πεδίο avaliable φαίνεται αν το μέρος είναι διαθέσιμο προς booking ή όχι ενώ το πεδίο approved είναι αυτό που περιέχει την πληροφορία για το εάν ο admin εγκρίνει ή όχι το οίκημα ούτως ώστε να δοθεί προς ενοικίαση. Αν η τιμή του είναι 0 τότε το οίκημα είναι ορατό από τους χρήστες διαφορετικά το οίκημα φαίνεται μόνο στους admins.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	p_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
2	owner	int(5)			No	None			Change Drop More
3	title	varchar(30)	utf8_general_ci		No	None			Change Drop More
4	description	varchar(500)	utf8_general_ci		No	None			Change Drop More
5	address	varchar(100)	utf8_general_ci		No	None			Change Drop More
6	city	int(5)			No	None			Change Drop More
7	bedrooms	int(3)			No	None			Change Drop More
8	bathrooms	int(3)			No	None			Change Drop More
9	max_people	int(3)			No	None			Change Drop More
10	price	float(10,2)			No	None			Change Drop More
11	available	tinyint(1)			No	1			Change Drop More
12	approved	tinyint(1)			No	0			Change Drop More

Εικόνα 11: Πεδία του πίνακα places

Ο πίνακας **place_has_features** έχει δημιουργηθεί όπως δηλώνει και το όνομα του για να πραγματοποιηθεί η σύνδεση μεταξύ των πινάκων places και features. Περιέχει τα πεδία phf_id, place, feature. Το πεδίο place είναι ξένο κλειδί από τον πίνακα places και το πεδίο feature είναι ξένο κλειδί από τον πίνακα features και περιέχουν τα μοναδικά ids της αντίστοιχης εγγραφής αυτών των πινάκων. Τα ξένα κλειδιά συνδέουν δύο πίνακες μεταξύ τους, και περιέχουν το primary key του πίνακα με τον οποίο θέλουν να επιτύχουν τη σύνδεση.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	phf_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
2	place	int(5)			No	None			Change Drop More
3	feature	int(5)			No	None			Change Drop More

Εικόνα 12: Πεδία του πίνακα place_has_features

Αντίστοιχα με τον πίνακα place_has_features έχει δημιουργηθεί και ο πίνακας **place_has_images**. Ένας πίνακας μπορεί να έχει πολλές εικόνες οπότε χρειάζεται κάτι παραπάνω από ένα πεδίο πίνακα για να αποθηκευτεί η πληροφορία. Η σχέση που συνδέει τον πίνακα place με μία εικόνα είναι Np1 οπότε θα έχουμε τη δημιουργία του πίνακα place_has_images αλλά όχι πίνακα images καθώς όλες οι εικόνες ενός place μπορούν να αποθηκευτούν εδώ. Τα πεδία του πίνακα είναι τα phi_id, place και url. Το πρώτο πεδίο είναι το ξεχωριστό id της κάθε εγγραφής, το δεύτερο πεδίο είναι ο κωδικός του place στο οποίο ανήκει η εικόνα που θα αποθηκευτεί με την μορφή url στο τρίτο και τελευταίο πεδίο του πίνακα.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	phi_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
2	place	int(5)			No	None			Change Drop More
3	url	varchar(500)	utf8_general_ci		No	None			Change Drop More

Εικόνα 13: Πεδία του πίνακα place_has_images

Τα **reviews** είναι ο πίνακας στον οποίο υπάρχουν οι πληροφορίες για τις αξιολογήσεις που μπορεί να αφήσει ένας χρήστης για ένα συγκεκριμένο μέρος. Ο πίνακας αυτός συνδέεται με τον πίνακα users με σχέση 1pN. Ένας χρήστης μπορεί να έχει πολλά reviews αλλά ένα review δεν μπορεί να ανήκει σε πολλούς διαφορετικούς χρήστες. Αντίστοιχα με τον πίνακα places όπου ένα place μπορεί να έχει πολλά διαφορετικά review αλλά το ίδιο review δεν μπορεί να ανήκει σε πολλά διαφορετικά places ταυτόχρονα. Οπότε τα πεδία user και place είναι ξένα κλειδιά που συνδέουν τον πίνακα reviews με τους πίνακες users και places αντίστοιχα. Το πεδίο rating περιέχει την βαθμολογία που αφήνει ένας χρήστης στο μέρος για το οποίο γίνεται η συγκεκριμένη εγγραφή και μπορεί να κυμαίνεται μεταξύ των βαθμών 1 και 5. τέλος το πεδίο text περιέχει ένα κείμενο το οποίο μπορεί να αφήσει ένας επισκέπτης ως σχόλιο.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	rev_id	int(5)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	user	int(5)			No	*None*			Change Drop More
☐	3	place	int(5)			No	*None*			Change Drop More
☐	4	rating	int(5)			No	*None*			Change Drop More
☐	5	text	varchar(5000)	utf8_general_ci		No	*None*			Change Drop More

Εικόνα 14: Πεδία του πίνακα reviews

Ο πίνακας **roles** περιέχει δύο πεδία το r_id και το name. Το πεδίο r_id είναι αυτόματα αυξανόμενο πεδίο και μοναδικό για κάθε εγγραφή και χρησιμοποιείται για να μπορεί να γίνει διαχωρισμός της κάθε εγγραφής. Το πεδίο name περιέχει το όνομα του ρόλου που υποστηρίζει η εφαρμογή. Η εφαρμογή περιέχει μόνο δύο ρόλους αυτόν του user και αυτόν του admin ωστόσο θα μπορούσαν να προστεθούν και επιπλέον ρόλοι.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	r_id	int(5)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(20)	utf8_general_ci		No	*None*			Change Drop More

Εικόνα 15: Πεδία του πίνακα roles

Στον πίνακα **users** αποθηκεύονται τα προσωπικά στοιχεία του κάθε χρήστη. Περιέχει τα πεδία u_ id, κύριο κλειδί που αυξάνεται με την κάθε εγγραφή, username που είναι το όνομα με το οποίο θέλει να φαίνεται ο χρήστης, password που είναι ο κωδικός που χρειάζεται ο χρήστης για να συνδεθεί με την εφαρμογή, email που είναι το email που δηλώνει ο χρήστης, f_name το όνομα του χρήστη, surname που είναι το επώνυμο του χρήστη και role που δείχνει αν ο χρήστης είναι απλός user ή admin. ο κάθε ρόλος έχει διαφορετικές δυνατότητες μέσα στην εφαρμογή για αυτό και είναι απαραίτητος ο διαχωρισμός των χρηστών με βάση αυτή τους την ιδιότητα.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	u_id	int(5)			No	None		AUTO_INCREMENT	Change Drop More
2	username	varchar(20)	utf8_general_ci		No	None			Change Drop More
3	password	varchar(50)	utf8_general_ci		No	None			Change Drop More
4	email	varchar(50)	utf8_general_ci		No	None			Change Drop More
5	f_name	varchar(20)	utf8_general_ci		No	None			Change Drop More
6	surname	varchar(20)	utf8_general_ci		No	None			Change Drop More
7	role	int(5)			No	None			Change Drop More

Εικόνα 16: Πεδία του πίνακα users

Η βάση που περιγράφηκε μόλις χρησιμοποιήθηκε κατασκευάζοντας κώδικα με τη χρήση των γλωσσών PHP και HTML. Ο κώδικας αυτός παρουσιάζεται στη συνέχεια και επεξηγείται εν συντομία. Όπως έχει αναφερθεί η HTML είναι υπεύθυνη για την κατασκευή της εμφάνισης μίας σελίδας ενώ η λειτουργικότητα της κατασκευάζεται με την PHP. Για αυτό το λόγο για κάθε σελίδα της εφαρμογής που θα παρουσιάζουμε θα δίνεται και ο κώδικας υλοποίησής της και στις δύο γλώσσες.

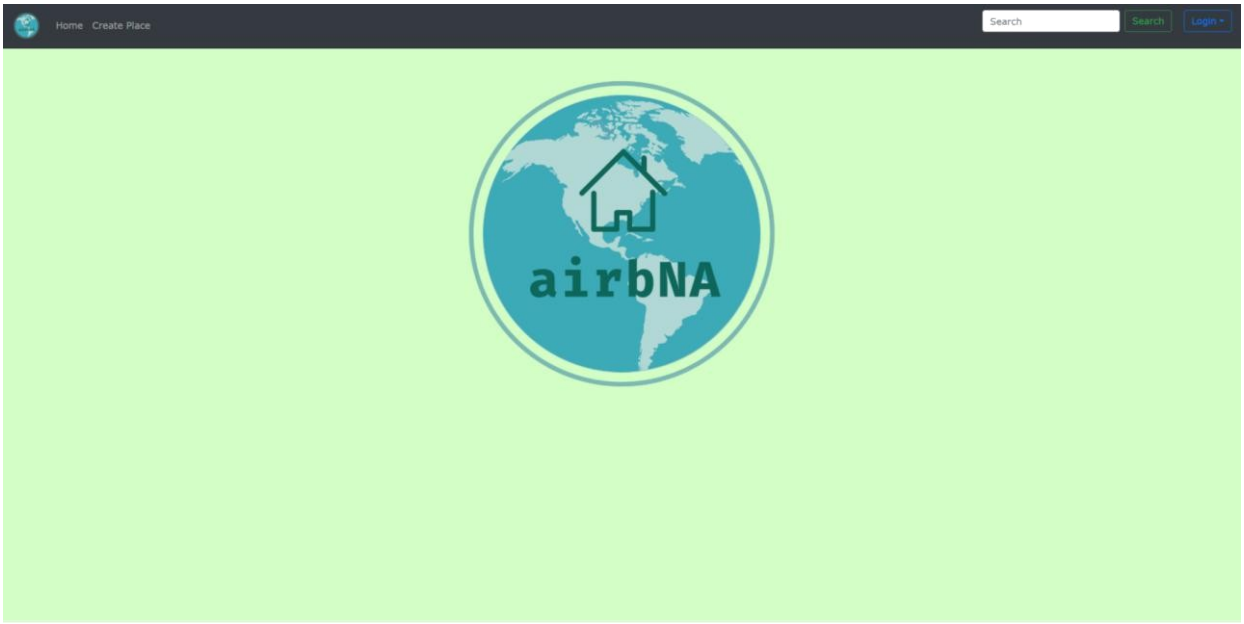
Configuration

Όπως αναφέρθηκε ήδη για την δημιουργία της εφαρμογής αλλά και για να εκτελεστούν οι λειτουργίες της έγινε χρήση βάσης τύπου MySQL. Για να συνδεθεί η βάση με την εφαρμογή και να πραγματοποιούνται λειτουργίες όπως η αποθήκευση δεδομένων από το χρήστη, η διαγραφή τους αλλά και η ενημέρωση ήδη υπαρκτών δεδομένων εκτελέστηκαν οι παρακάτω εντολές που αποτελούν και τη διαδικασία που ονομάζουμε configuration.

```php
<?php

define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'airbnb');

$link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
```

Εικόνα 17: Κώδικας του αρχείου config.php

Index



Εικόνα 18: Η αρχική σελίδα της εφαρμογής PHP

Αυτή είναι η αρχική σελίδα που βλέπουν όλοι οι χρήστες της εφαρμογής τη στιγμή που μπαίνουν στο site.

```
index.php > ...
1   <html>
2   <head>
3       <?php include 'includes/tools.html';?>
4   </head>
5
6   <body>
7
8   <?php include 'includes/navbar.php';?>
9
10  <img class="logo" src="media/logo.png" height="600px" width="600px">
11
12  </body>
13  </html>
```

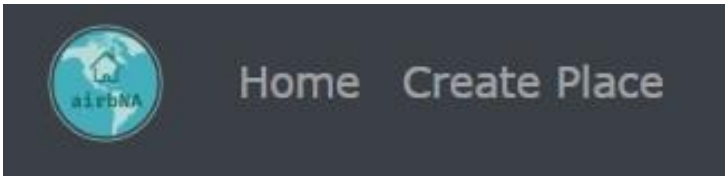
Εικόνα 19: Κώδικας του αρχείου index.php

Στη γραμμή τρία βλέπουμε ότι γίνεται εισαγωγή κάποιων εργαλείων “tools” που περιέχουν απαραίτητα script και βιβλιοθήκες τα οποία χρησιμοποιούνται για διάφορες λειτουργίες της εφαρμογής όπως ο καθορισμός της εμφάνισης των σελίδων με συγκεκριμένο styling.

```
includes > <> tools.html > ...
1   <meta charset="UTF-8">
2   <title>AirbNA</title>
3
4   <link rel="stylesheet" href="includes/styles.css">
5   <link rel="stylesheet" href="includes/external/bootstrap.min.css">
6   <link rel="stylesheet" href="includes/external/w3.css">
7   <!--link rel="stylesheet" href="includes/external/font-awesome.min.css"-->
8   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css">
9
10  <script src="includes/external/jquery-3.3.1.slim.min.js"></script>
11  <script src="includes/external/popper.min.js"></script>
12  <script src="includes/external/bootstrap.min.js"></script>
13
14  <link rel="icon" href="media/logo.png" sizes="16x16" type="image/jpg">
15
16  <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
17  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
18
19  <script>
20      $( function() {
21          $( "#datepicker" ).datepicker();
22      } );
23  </script>
```

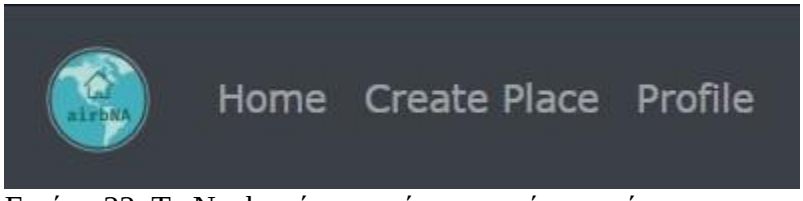
Εικόνα 20: Κώδικας του αρχείου tools.html

Στο πάνω κομμάτι της σελίδας υπάρχει το navbar και στη γραμμή 8 του αρχείου index εισάγουμε το navbar στη σελίδα ούτως ώστε να εμφανίζεται στους χρήστες. Το navbar είναι η βοηθητική μπάρα που εμφανίζεται σε όλες τις σελίδες της εφαρμογής και αποτελεί το menu μετάβασης στις υπόλοιπες σελίδες όπως και το search και login. Στην αριστερή πλευρά του navbar υπάρχει το logo της εφαρμογής καθώς και οι επιλογές Home και Create Place που μπορεί να επιλέξει ο χρήστης.



Εικόνα 21: Το Navbar όπως φαίνεται σε έναν επισκέπτη

Σημειώνεται ότι η εμφάνιση του navbar έχει αυτή τη μορφή όταν δεν υπάρχει συνδεδεμένος χρήστη στην εφαρμογή. Αν ο χρήστης είναι συνδεδεμένος και έχει πάρει το ρόλο User τότε το αριστερό κομμάτι του navbar διαμορφώνεται εμφανισιακά στη σελίδα ως εξής:



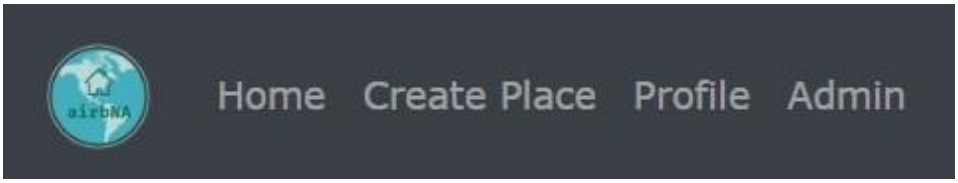
Εικόνα 22: Το Navbar όπως φαίνεται σε έναν χρήστη

Για τη δημιουργία του navbar γράφτηκε ο παρακάτω κώδικας:

```
12      <div class="collapse navbar-collapse" id="navbarSupportedContent">
13          <ul class="navbar-nav mr-auto">
14              <li class="nav-item">
15                  <a class="nav-link" href="index.php">Home</a>
16              </li>
17              <?php
18              if (isset($_SESSION['logged_user'])) {
19                  echo "<li class=\"nav-item\">";
20                  echo "<a class=\"nav-link\" href=\"create_place.php\">Create Place</a>";
21                  echo "</li>";
22              } else {
23                  echo "<li class=\"nav-item\">";
24                  echo "<a class=\"nav-link\" href=\"register.php\">Create Place</a>";
25                  echo "</li>";
26              }
27              ?>
28              <?php
29              if (isset($_SESSION['logged_user'])) {
30                  echo "<li class=\"nav-item\">";
31                  echo "<a class=\"nav-link\" href=\"profile.php\">Profile</a>";
32                  echo "</li>";
33              }
34              ?>
35              <?php
36              if (isset($_SESSION['user_role']) && $_SESSION['user_role'] == 'admin') {
37                  echo "<li class=\"nav-item\">";
38                  echo "<a class=\"nav-link\" href=\"admin.php\">Admin</a>";
39                  echo "</li>";
40              }
41              ?>
42          </ul>
```

Εικόνα 23: Ο κώδικας δημιουργίας του Navbar

Στις γραμμές 18-25 ο κώδικας αναφέρεται στο κομμάτι Create Place. Με τη χρήση session ελέγχεται αν υπάρχει συνδεδεμένος χρήστης στην εφαρμογή αυτή τη στιγμή ούτως ώστε να μεταβεί στη σελίδα Create Place διαφορετικά αν δεν υπάρχει κάποιος χρήστης που να έχει πραγματοποιήσει Login τότε εάν επιλέξει την επιλογή Create Place θα μεταβεί στη σελίδα Register. Στις γραμμές 29-32 ελέγχεται και πάλι με τη χρήση session αν υπάρχει συνδεδεμένος χρήστης και μόνο εάν υπάρχει ο χρήστης επιλέγοντας την επιλογή Profile θα μεταβεί στο profile του. Στις γραμμές 36-39 πέραν του ότι ελέγχεται αν υπάρχει συνδεδεμένος χρήστης ελέγχεται και εάν ο χρήστης αυτός είναι ο admin του συστήματος και εφόσον πληρεί αυτές τις προϋποθέσεις μπορεί μέσο της επιλογής admin να μεταβεί στην σελίδα admin. Εάν ο χρήστης έχει ρόλο admin τότε μετά το login του το navbar θα του δίνει μία επιπλέον επιλογή αυτή του admin.



Εικόνα 24: Το Navbar όπως φαίνεται σε έναν admin

Τελευταίο στοιχείο που αξίζει να σημειωθεί είναι το logo της εφαρμογής το οποίο είναι πάντοτε ορατό στην αριστερή πλευρά του navbar και πατώντας το ο χρήστης μπορεί να μεταβεί από οποιαδήποτε σελίδα βρίσκεται στην αρχική σελίδα index της εφαρμογής.

```
<nav class="navbar sticky-top navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="index.php"><img class="logo" src="media/logo.png" height="50px" width="50px"></a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
```
Εικόνα 25: Ο κώδικας εμφάνισης του logo

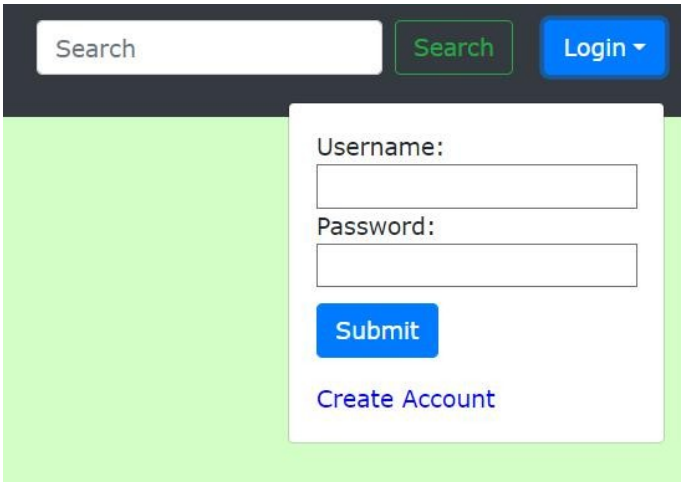
Στη δεξιά πλευρά του navbar εμφανίζονται το πεδίο search και οι επιλογές Search και Login.

Εικόνα 26: Οι επιλογές Search και Login του Navbar

Ο κώδικας της μορφοποίησης για την δεξιά πλευρά του navbar είναι ο εξής.

```
        <ul class="nav pull-right">
            <!--searchbar -->
            <li class='nav-item'>
                <form class="form-inline" name='searchform' method='post'>
                    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search" name="search">
                    <input class="btn btn-outline-success my-2 my-sm-0" type="submit" name="srch" value="Search">
                </form>
            </li>
            <!-- log in -->
            <?php
            if (!isset($_SESSION['logged_user'])) {
                echo "<li class='dropdown nav-item' id='menuLogin'>";
                echo "<a style=\"margin-left: 20px;\" class='dropdown-toggle btn btn-outline-primary my-2 my-sm-0' href='#\' data-toggle='dropdown' id='navLogin'>Login</a>";
                echo "<div class='login-menu dropdown-menu' style='padding:17px;'>";
                echo "<form class='form-inline' action='' name='login' method='post'>";
                echo "<label>Username: </label> <input type='text' name='user'><br>";
                echo "<label>Password: </label> <input type='password' name='pass'><br>";
                echo "<input style='margin-top: 10px;' type='submit' class='btn btn-primary' value='Submit' name='logsbmt'>";
                echo "</form>";
                echo "<a href='register.php'>Create Account</a>";
                echo "</div>";
                echo "</li>";
            }
            ?>
            <!-- log out -->
            <?php
            if (isset($_SESSION['logged_user'])) {
                echo "<li class='nav-item'>";
                echo "<form action='' name='logout' method='post'>";
                echo "<input style=\"margin-left: 20px;\" class='btn btn-outline-danger my-2 my-sm-0' type='submit' value='Logout' name='logout'>";
                echo "</form>";
                echo "</li>";
            }
            ?>
        </ul>
    </div>
</nav>
```
Εικόνα 27: Ο κώδικας δημιουργίας των επιλογών Search και Login

Όπως φαίνεται με χρήση sessions ελέγχεται εάν ο χρήστης είναι συνδεδεμένος η όχι. Αν δεν είναι τότε βλέπει την επιλογή login που είναι μία drop down list και και περιέχει τα πεδία που χρειάζεται να συμπληρώσει ο χρήστης για να συνδεθεί στην εφαρμογή ενώ αν είναι συνδεδεμένος τότε απλά του εμφανίζεται η επιλογή log out που επιλέγοντας την αποσυνδέεται από την εφαρμογή.



Εικόνα 28: Το παράθυρο Login

Στο παραπάνω σχήμα φαίνεται το πλαίσιο που εμφανίζεται στον χρήστη εαν επιλέξει την επιλογή login. Ο κώδικας λειτουργικότητας είναι ο εξής:

```php
// Login
if (isset($_POST['logsbmt'])) {
    extract($_POST);
    if ($user && $pass) {

        $resultlog = mysqli_query($link, "SELECT * FROM users WHERE username = '$user' AND password = '$pass'");

        if (!$resultlog) {
            printf("Error during reading.\n");
        } else {
            while ($row = $resultlog->fetch_assoc()) {
                unset($user_id, $role_id);
                // Check for role
                if ($row['u_id'] && $row['role']) {
                    $user_id = $row['u_id'];
                    $role_id = $row['role'];

                    $result1 = mysqli_query($link, "SELECT name FROM roles WHERE r_id = '$role_id'");
                    while ($row1 = $result1->fetch_assoc()) {
                        unset($role);
                        $role = $row1['name'];
                    }
                    // Set role in session
                    if ($role == 'admin') {
                        unset($_SESSION['user_role']);
                        $_SESSION['user_role'] = 'admin';
                    } else {
                        unset($_SESSION['user_role']);
                        $_SESSION['user_role'] = 'user';
                    }
                    // Set user in session
                    unset($_SESSION['logged_id']);
                    $_SESSION['logged_id'] = $user_id;

                    unset($_SESSION['logged_user']);
                    $_SESSION['logged_user'] = $user;

                    mysqli_close($link);
                    header("Refresh:0");
                } else {
                    printf("Wrong Credentials.\n");
                }
            }
        }
    } else {
        echo "enter username and password";
    }
```

Εικόνα 29: Ο κώδικας υλοποίησης της λειτουργίας Login

Εδώ ουσιαστικά με χρήση κατάλληλων queries γίνεται αναζήτηση στη βάση για τον αν υπάρχει καταχώρηση με τα στοιχεία που έχει δώσει ο χρήστης στα πεδία του login ενώ αποθηκεύεται και ο ρόλος που έχει ο συγκεκριμένος χρήστης σε κατάλληλο session ούτως ώστε να κρατηθεί η πληροφορία και να χρησιμοποιηθεί όπως είδαμε είτε για τις επιλογές που θα του εμφανιστούν στο navbar στη συγκεκριμένη σελίδα είτε για λόγους λειτουργικότητας επόμενων σελίδων που θα παρουσιαστούν παρακάτω. Αντίστοιχα στο logout αποδεσμεύονται τα στοιχεία του χρήστη που έχουμε αποθηκεύσει με session και ο χρήστης μεταφέρεται στην αρχική σελίδα index.

```php
// Logout
if (isset($_POST['logout'])) {
    unset($_SESSION['logged_user']);
    unset($_SESSION['user_role']);
    header("location: index.php");
}
```

Εικόνα 30: Ο κώδικας υλοποίησης της λειτουργίας Logout

Το πεδίο search είναι τύπου text και εκεί ο χρήστης μπορεί να πραγματοποιήσει αναζήτηση με βάση την ονομασία ενός μέρους ή την πόλη στην οποία θέλει να αναζητήσει ένα μέρος. Αφού ο χρήστης βάλει τα στοιχεία αυτά για να πραγματοποιηθεί η αναζήτησή του πρέπει να πατήσει το κουμπί search. Αν ο τίτλος ή η πόλη υπάρχουν στη βάση από την οποία παίρνουμε στοιχεία με την εκτέλεση του κατάλληλου query τότε ο χρήστης θα μεταβεί αυτόματα σε άλλη σελίδα στην οποία του εμφανίζονται τα αποτελέσματα της αναζήτησής του διαφορετικά ο χρήστης και πάλι θα μεταβεί σε σελίδα εμφάνισης αποτελεσμάτων αλλά η θέση των αποτελεσμάτων θα είναι κενή. Ακολουθεί ο κώδικας υλοποίησης της διαδικασίας που μόλις περιγράφηκε.

```php

// Search
if (isset($_POST['srch'])) {
    extract($_POST);

    $results = array();

    if ($search) {
        $searchvar = "%" . $search . "%";
        $result2 = mysqli_query($link, "SELECT p_id FROM places WHERE (city = (SELECT c_id FROM cities WHERE name = '$search') OR title LIKE '$searchvar') AND approved = 1");

        while ($row2 = $result2->fetch_assoc()) {
            array_push($results, $row2['p_id']);
        }

        $_SESSION['search_results'] = $results;

        mysqli_close($link);
        header("location: search_results.php");
    }
}

?>
```

Εικόνα 31: Ο κώδικας υλοποίησης της λειτουργίας Search

Αξίζει να σημειωθεί ότι το query που εκτελείται και ελέγχει αν αυτά που έχει δώσει ο χρήστης ταυτίζονται με κάποια εγγραφή στη βάση δεδομένων ελέγχει ταυτόχρονα και αν το μέρος που θα εμφανίσει είναι approved από τους Admin του συστήματος.

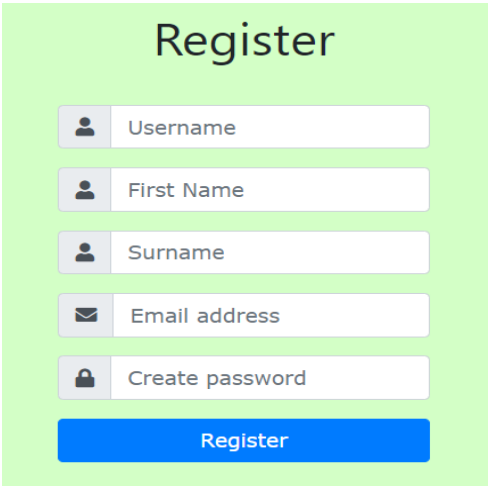
Register

Στην περίπτωση που ο χρήστης προσπαθήσει να πραγματοποιήσει την ενέργεια create playce ή την ενέργεια create account που εμφανίζεται στην οθόνη όταν επιλέγεται η ενέργεια Login τότε θα μεταβεί στην σελίδα Register. Η σελίδα αυτή έχει δημιουργηθεί ούτως ώστε ο χρήστης εισάγοντας τα προσωπικά στοιχεία που θα του ζητηθούν από τη φόρμα να δημιουργήσει λογαριασμό στην εφαρμογή. Με την έννοια δημιουργία λογαριασμού εννοούμαι ότι αποθηκεύονται στη βάση τα στοιχεία του χρήστη ούτως ώστε αυτά να μπορούν να ανατηχθούν κάθε φορά που αυτός συνδέεται στην εφαρμογή και θέλει να τη χρησιμοποιήσει. Για να αποθηκευτούν αυτά τα στοιχεία στη βάση αλλά και να εμφανιστούν τα πεδία συμπλήρωσης στο χρήστη εκτελείται ο παρακάτω κώδικας:

```
73  if (isset($_POST['sbmt'])) {
74
75      extract($_POST);
76
77      if ($user && $password && $email && $fname && $sname) {
78          $result = mysqli_query($link, "SELECT username, email FROM users WHERE username='$user' OR email='$email'");
79          if ($result->fetch_assoc()) {
80              echo "username or email already exists";
81          } else {
82              $result1 = mysqli_query($link, "INSERT INTO users (username, password, email, f_name, surname, role) VALUES ('$user', '$password', '$email', '$fname', '$sname', 2)");
83              if (!$result1) printf("Error during insertion.\n");
84              else {
85                  header("location: index.php");
86                  mysqli_close($link);
87              }
88          }
89      }
90  }
```

Εικόνα 32: Ο κώδικας υλοποίησης της λειτουργίας Register

Στη γραμμή 77 ελέγχεται αν ο χρήστης έχει συμπληρώσει όλα τα πεδία της φόρμας και αν αυτό ισχύει τότε εκτελείται το query της γραμμής 78 που ελέγχει αν υπάρχει ήδη κάποιος χρήστης με κοινό username ή email αν δεν υπάρχει χρήστης με αυτά τα στοιχεία τότε ο πίνακας users ενημερώνεται εισάγοντας νέο χρήστη. Στην περίπτωση που δεν έχουν συμπληρωθεί όλα τα πεδία της φόρμας τότε δεν γίνεται εγγραφή στον πίνακα users και ο χρήστης πρέπει να προσπαθήσει με νέα καταχώρηση των στοιχείων του στη φόρμα.



Εικόνα 33: Η φόρμα Register

Για να δημιουργηθεί αυτό το πλαίσιο χρησιμοποιήθηκε ο εξής κώδικας:

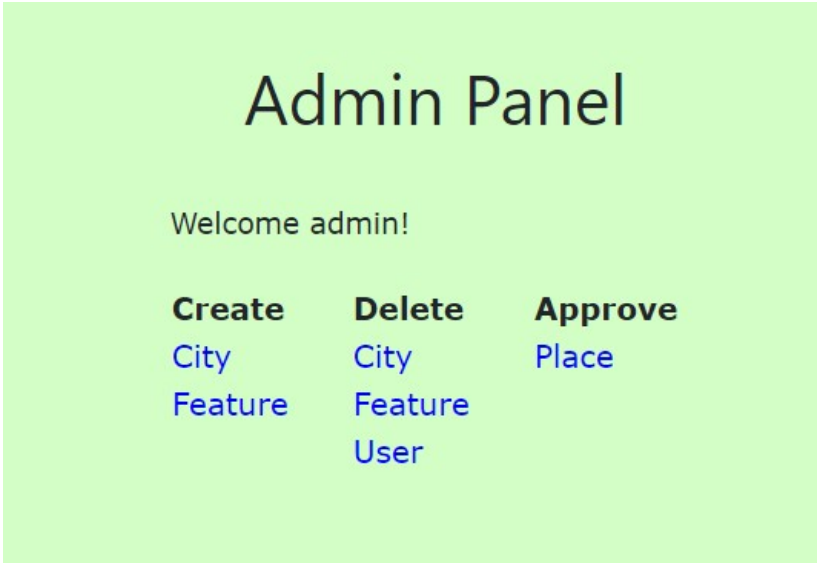
```
<div class="card">
    <article class="card-body mx-auto" style="max-width: 400px;">
        <h1 class="card-title mt-3 text-center">Register</h1><br>
        <form name="register" method="post" action="">
            <!-- Username Field -->
            <div class="form-group input-group">
                <div class="input-group-prepend">
                    <span class="input-group-text"> <i class="fas fa-user"></i> </span>
                </div>
                <input name="user" class="form-control" placeholder="Username" type="text">
            </div>
            <!-- First Name Field -->
            <div class="form-group input-group">
                <div class="input-group-prepend">
                    <span class="input-group-text"> <i class="fa fa-user"></i> </span>
                </div>
                <input name="fname" class="form-control" placeholder="First Name" type="text">
            </div>
            <!-- Surname Field -->
            <div class="form-group input-group">
                <div class="input-group-prepend">
                    <span class="input-group-text"> <i class="fa fa-user"></i> </span>
                </div>
                <input name="sname" class="form-control" placeholder="Surname" type="text">
            </div>
            <!-- Email Field -->
            <div class="form-group input-group">
                <div class="input-group-prepend">
                    <span class="input-group-text"> <i class="fa fa-envelope"></i> </span>
                </div>
                <input name="email" class="form-control" placeholder="Email address" type="email">
            </div>
            <!-- Password Field -->
            <div class="form-group input-group">
                <div class="input-group-prepend">
                    <span class="input-group-text"> <i class="fa fa-lock"></i> </span>
                </div>
                <input name="password" class="form-control" placeholder="Create password" type="password">
            </div>
            <!-- Submit Button -->
            <div class="form-group">
                <button type="submit" name="sbmt" class="btn btn-primary btn-block"> Register </button>
            </div>
        </form>
    </article>
</div>
```

Εικόνα 34: Ο κώδικας δημιουργίας της φόρμας Register

Στην ουσία δημιουργείται μία “card” που περιέχει τη φόρμα με τίτλο register και στη φόρμα αυτή υπάρχουν τα text πεδία στα οποία θέλουμε να συμπληρώσει ο χρήστης τις πληροφορίες που του ζητούνται. Τα πεδία αυτά ομαδοποιούνται μεταξύ τους ενώ με χρήση κατάλληλων εντολών παίρνουν και την τελική μορφή την οποία τελικά θα δει ο χρήστης στην οθόνη του.

Admin Panel

Αν επιλεχθεί στη σελίδα index η επιλογή admin που δίνεται στο navbar και που μόνο ένας χρήστης που έχει συνδεθεί ως admin μπορεί να δει και να επιλέξει τότε η εφαρμογή πραγματοποιεί μετάβαση στη σελίδα Admin Panel. Στη σελίδα αυτή ο χρήστης έχει μπροστά του ένα μενού επιλογών με τη μορφή links που επιλέγοντας τα μεταβαίνει και στις ανάλογες σελίδες ούτως ώστε να πραγματοποιήσει την ενέργεια που θέλει. Η σελίδα έχει την εξής μορφή:



Εικόνα 35: Η σελίδα Admin Panel

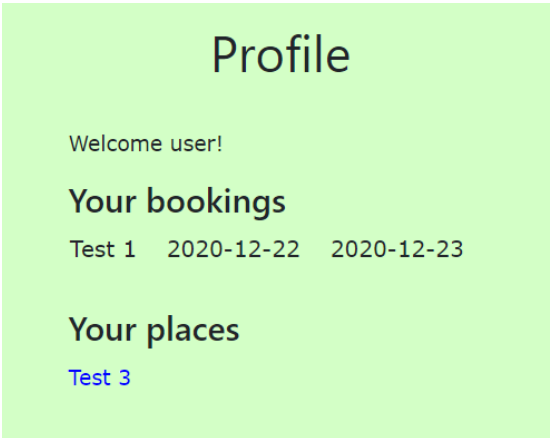
Και αν για παράδειγμα ο admin θέλει να κάνει approve κάποιο μέρος τότε θα πρέπει να πάει στη στήλη Approve και να επιλέξει το link Place το οποίο και θα τον μεταφέρει στην κατάλληλη σελίδα ούτως ώστε να κάνει approve το μέρος που θέλει. Αντίστοιχα λειτουργούν και οι υπόλοιπες επιλογές του panel. Για να πραγματοποιηθούν όλες αυτές οι μεταβάσεις από το panel στις άλλες σελίδες της εφαρμογής αλλά και για να εμφανιστεί το panel στον admin εκτελέστηκε ο ακόλουθος κώδικας:

```
17      <div class="card">
18          <article class="card-body mx-auto" style="max-width: 400px;">
19              <h1 class="card-title mt-3 text-center">Admin Panel</h1><br>
20              <div class="content">
21                  Welcome <?php if (isset($_SESSION['logged_user'])) {
22                              echo $_SESSION['logged_user'];
23                          } ?>!<br><br>
24                  <table class="admin-table">
25                      <thead style="font-weight: bold;">
26                          <td>Create</td>
27                          <td>Delete</td>
28                          <td>Approve</td>
29                      </thead>
30                      <tr>
31                          <td><a href="create_city.php">City</a></td>
32                          <td><a href="delete_city.php">City</a></td>
33                          <td><a href="approve_place.php">Place</a></td>
34                      </tr>
35                      <tr>
36                          <td><a href="create_feature.php">Feature</a></td>
37                          <td><a href="delete_feature.php">Feature</a></td>
38                      </tr>
39                      <tr>
40                          <td></td>
41                          <td><a href="delete_user.php">User</a></td>
42                      </tr>
43                  </table>
44
45              </div>
```

Εικόνα 36: Ο κώδικας δημιουργίας της σελίδας Admin Panel

User Profile

Επιλέγοντας την επιλογή profile από το navbar ένας user ή ένας admin της εφαρμογής μπορεί να μεταβεί στη σελίδα όπου παρουσιάζεται το profile του. Αυτό έχει την παρακάτω μορφή.



Εικόνα 37: Η σελίδα Profile

Βλέπει ένα μήνυμα καλωσορίσματος στην σελίδα και μετά παρουσιάζονται πληροφορίες για τα bookings που έχει πραγματοποιήσει (όνομα οικήματος, ημερομηνία άφιξης και ημερομηνία αποχώρησης) αλλά και λίστα με τη μορφή links που περιέχει τα οικήματα που ανήκουν στον συγκεκριμένο user. Επιλέγοντας ένα οίκημα ο χρήστης μεταβαίνει στην σελίδα με τις πληροφορίες του συγκεκριμένου οικήματος (view place) τα κομμάτια κώδικα που πραγματοποιούν τη σύνδεση με τη βάση και αντλούν πληροφορίες τόσο για τα bookings του χρήστη όσο και για τα places του δίνονται παρακάτω.

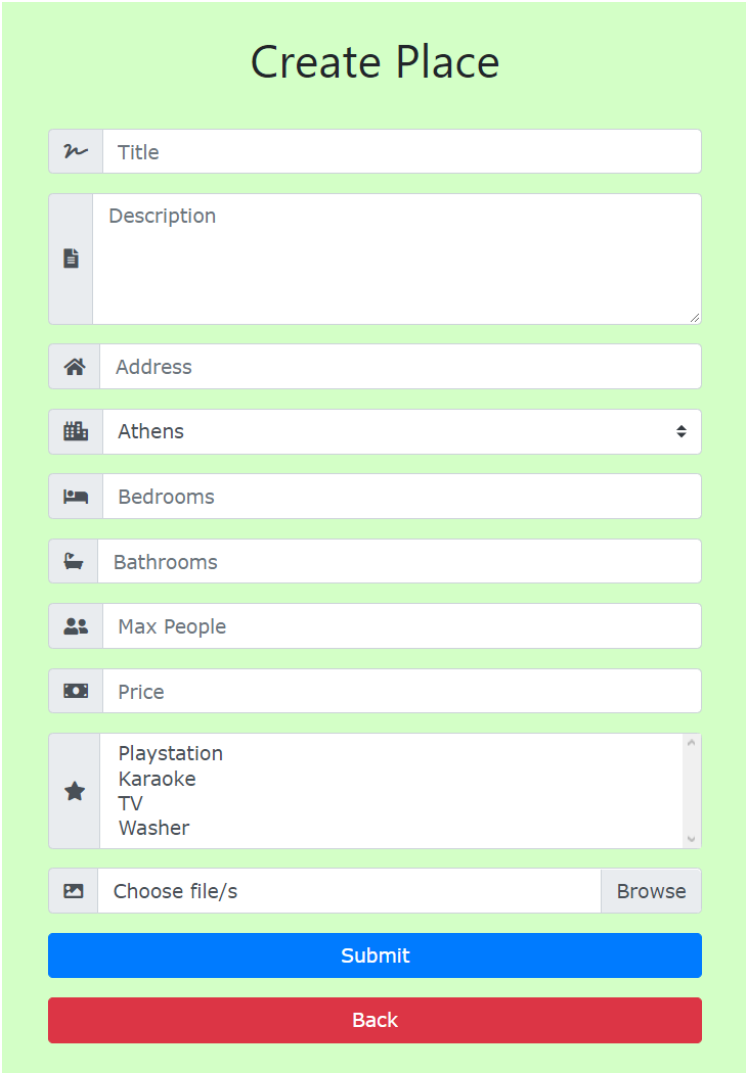
```
<p>Welcome <?php echo $_SESSION['logged_user']; ?>!</p>

<h3>Your bookings</h3>
<table class="bookings-table">
    <?php
    require_once "config.php";
    $user = $_SESSION['logged_user'];
    $result = mysqli_query($link, "SELECT places.title, bookings.b_id, bookings.start_date, bookings.end_date FROM bookings, places WHERE bookings.user = (SELECT u_id FROM users WHERE
    username = '$user') && (bookings.place = places.p_id)");
    while ($row = $result->fetch_assoc()) {
        unset($id, $title, $start_date, $end_date);
        $id = $row['b_id'];
        $title = $row['title'];
        $start_date = $row['start_date'];
        $end_date = $row['end_date'];

        echo '<tr>
        <td>' . $title . '</td>
        <td>' . $start_date . '</td>
        <td>' . $end_date . '</td>
        </tr>';
    }
    ?>
</table>
<br>
```

Εικόνα 38: Ο κώδικας εμφάνισης των bookings του χρήστη

```
<h3>Your places</h3>
<ul>
    <?php
    require_once "config.php";
    $user = $_SESSION['logged_user'];
    $result2 = mysqli_query($link, "SELECT p_id, title FROM places WHERE owner = (SELECT u_id FROM users WHERE username = '$user')");
    while ($row2 = $result2->fetch_assoc()) {
        unset($id2, $title2);
        $id2 = $row2['p_id'];
        $title2 = $row2['title'];
        echo '<a href="place_info.php?id=' . $id2 . '">' . $title2 . '</a><br>';
    }
    ?>
</ul>
```

Εικόνα 39: Ο κώδικας εμφάνισης των places του χρήστη

Create Place



Εικόνα 40: Η φόρμα δημιουργίας νέου place

Αυτή είναι η δομή της σελίδας Create Place. Όπως δηλώνει και το όνομα της εδώ θα κατευθυνθούν οι χρήστες της εφαρμογής που θέλουν να δημιουργήσουν ένα οίκημα ούτως ώστε αυτό να μπορεί να διατεθεί προς ενοικίαση. Απαιτείται από τον χρήστη η συμπλήρωση όλων των πεδίων της φόρμας ούτως ώστε οι πληροφορίες αυτές να αποθηκευτούν τελικά στον πίνακα place. Για να πραγματοποιηθεί η εισαγωγή των στοιχείων ο χρήστης πρέπει να επιλέξει το κουμπί submit ενώ με το πάτημα του κουμπιού back ο χρήστης επιστρέφει στην αρχική σελίδα. Αξίζει να σημειωθεί ότι το πεδίο cities έχει προεπιλεγμένη τιμή Athens και επιλέγοντας το ο χρήστης εμφανίζεται μία drop down list στην οποία περιέχονται όλες οι πόλεις της εφαρμογής. Η υλοποίηση αυτή επιλέχθηκε για να αποφευχθούν λάθη που έχουν να κάνουν με την εισαγωγή πόλεων από τους χρήστες οι οποίες όμως δεν υπάρχουν στα δεδομένα της εφαρμογής. Επίσης το πεδίο στο οποίο εμφανίζονται τα features έχει διαφορετικό τρόπο υλοποίησης από τα υπόλοιπα πεδία. Στον χρήστη εμφανίζεται μία λίστα με όλα τα features που

υποστηρίζει η εφαρμογή και μπορεί να επιλέξει όσα από αυτά θέλει να δηλώσει ότι μπορεί να βρει κάποιος στο οίκημα του αν επιλέξει να το νοικιάσει. Τέλος στο πεδίο με τις εικόνες ο χρήστης μπορεί να ανεβάσει πλήθος αρχείων εικόνων από τη συσκευή του και στο τέλος θα εμφανίζεται το τελευταίο και μόνο αρχείο που επέλεξε στο πεδίο αυτό.

```
46                  <!-- City -->
47                  <div class="form-group input-group">
48                      <div class="input-group-prepend">
49                          <label class="input-group-text" for="inputGroupSelect01"><i class="fas fa-city"></i></label>
50                      </div>
51                      <select name="city" class="custom-select" id="inputGroupSelect01">
52                          <?php
53                          require_once "config.php";
54                          $result1 = mysqli_query($link, "SELECT * FROM cities");
55                          while ($row = $result1->fetch_assoc()) {
56                              unset($id, $name);
57                              $id = $row['c_id'];
58                              $name = $row['name'];
59                              echo '<option value="' . $id . '">' . $name . '</option>';
60                          }
61                          ?>
62                      </select>
63                  </div>
```

Εικόνα 41: Ο κώδικας δημιουργίας της drop down λίστας για τα cities

```
97                  <!-- Features -->
98                  <div class="form-group input-group">
99                      <div class="input-group-prepend">
100                         <label class="input-group-text" for="inputGroupSelect02"><i class="fas fa-star"></i></label>
101                     </div>
102                     <select multiple name="feature[]" class="custom-select" id="inputGroupSelect02">
103                         <?php
104                         $result1 = mysqli_query($link, "SELECT * FROM features");
105                         while ($row = $result1->fetch_assoc()) {
106                             unset($id, $name);
107                             $id = $row['f_id'];
108                             $name = $row['name'];
109                             echo '<option value="' . $id . '">' . $name . '</option><br>';
110                         }
111                         ?>
112                     </select>
113                 </div>
```

Εικόνα 42: Ο κώδικας δημιουργίας της λίστας για τα features

```
115                 <!-- Images -->
116                 <div class="form-group input-group">
117                     <div class="input-group-prepend">
118                         <span class="input-group-text"> <i class="fas fa-image"></i> </span>
119                     </div>
120                     <div class="custom-file">
121                         <input type="file" multiple class="custom-file-input" id="files" name="files[]" aria-describedby="inputGroupFileAddon01">
122                         <label class="custom-file-label" for="files[]">Choose file/s</label>
123                     </div>
124                 </div>
```

Εικόνα 43: Ο κώδικας δημιουργίας της επιλογής ανάρτησης εικόνας

```
    <script>
    $('#files').on('change', function() {
        //get the file name
        var fileName = $(this).val();
        var cleanFileName = fileName.replace('C:\\fakepath\\', " ");
        //replace the "Choose a file" label
        $(this).next('.custom-file-label').html(cleanFileName);
    })
</script>
```

Εικόνα 44: Ο κώδικας αλλαγής της ετικέτας της επιλογής ανάρτησης εικόνας

```
// File upload configuration
$targetDir = "uploads/";
$allowTypes = array('jpg', 'png', 'jpeg', 'gif', 'JPG', 'PNG', 'JPEG', 'GIF');

$statusMsg = $errorMsg = $image_url = $errorUpload = $errorUploadType = '';
if (!empty(array_filter($_FILES['files']['name']))) {
    foreach ($_FILES['files']['name'] as $key => $val) {
        // File upload path
        $fileName = basename($_FILES['files']['name'][$key]);
        $targetFilePath = $targetDir . $fileName;

        // Check whether file type is valid
        $fileType = pathinfo($targetFilePath, PATHINFO_EXTENSION);
        if (in_array($fileType, $allowTypes)) {
            // Upload file to server
            if (move_uploaded_file($_FILES["files"]["tmp_name"][$key], $targetFilePath)) {
                // Image db insert sql
                $image_url .= "(" . $last_id . ", 'uploads/" . $fileName . "'),";
            } else {
                $errorUpload .= $_FILES['files']['name'][$key] . ', ';
            }
        } else {
            $errorUploadType .= $_FILES['files']['name'][$key] . ', ';
        }
    }

    if (!empty($image_url)) {
        $image_url = trim($image_url, ',');
        // Insert image file name into database
        $result_img = mysqli_query($link, "INSERT INTO place_has_images (place, url) VALUES $image_url");
        if ($result_img) {
            $errorUpload = !empty($errorUpload) ? 'Upload Error: ' . $errorUpload : '';
            $errorUploadType = !empty($errorUploadType) ? 'File Type Error: ' . $errorUploadType : '';
            $errorMsg = !empty($errorUpload) ? '<br/>' . $errorUpload . '<br/>' . $errorUploadType : '<br/>' . $errorUploadType;
            $statusMsg = "Files are uploaded successfully." . $errorMsg;
        } else {
            $statusMsg = "Sorry, there was an error uploading your file.";
        }
    }
} else {
    $statusMsg = 'Please select a file to upload.';
}
```

Εικόνα 45: Ο κώδικας υλοποίησης της λειτουργίας ανάρτησης πολλαπλών εικόνων

Αξίζει επίσης να αναφερθεί ο τρόπος με των οποίο αποθηκεύονται οι εικόνες και τα features που έχουν επιλεχθεί στη βάση δεδομένων. Όπως παρουσιάστηκε ήδη όταν αναλύθηκε η δομή της βάσης οι εικόνες αλλά και το τι features έχει το κάθε place αποθηκεύονται σε ξεχωριστούς πίνακες και όχι στον ίδιο πίνακα place καθώς μπορεί να είναι περισσότερα από ένα δεδομένα αυτά που θα πρέπει να αποθηκευτούν πράγμα αδύνατο να πραγματοποιηθεί από ένα μόνο πεδίο. Τα queries που εκτελούν τις εγγραφές των στοιχείων στους αντίστοιχους πίνακες είναι αυτά που φαίνονται στην παρακάτω εικόνα.

```php
if ($title && $address && $price) {

    $result2 = mysqli_query($link, "INSERT INTO places (owner, title, description, address, city, bedrooms, bathrooms, max_people, price) VALUES ('$user', '$title', '$description', '$address',
    '$city', '$bedrooms', '$bathrooms', '$max_people', '$price')");

    $result_id = mysqli_query($link, "SELECT p_id FROM places ORDER BY p_id DESC LIMIT 1");
    $row = mysqli_fetch_row($result_id);
    $last_id = $row[0];

    if (isset($feature)) {
        $sql = "INSERT INTO place_has_features (place, feature) VALUES ";

        for ($i = 0; $i < sizeof($feature); $i++) {
            $sql .= "(" . $last_id . ", " . $feature[$i] . "), ";
        }
        $sql = substr(trim($sql), 0, -1);

        $result3 = mysqli_query($link, $sql);
    }
```

Εικόνα 46: Ο κώδικας υλοποίησης των query για τη δημιουργία place

Search Results

Όταν ένας χρήστης πραγματοποιεί τη λειτουργία search τότε τα αποτελέσματα αναζήτησης του θα εμφανιστούν στη σελίδα search results.

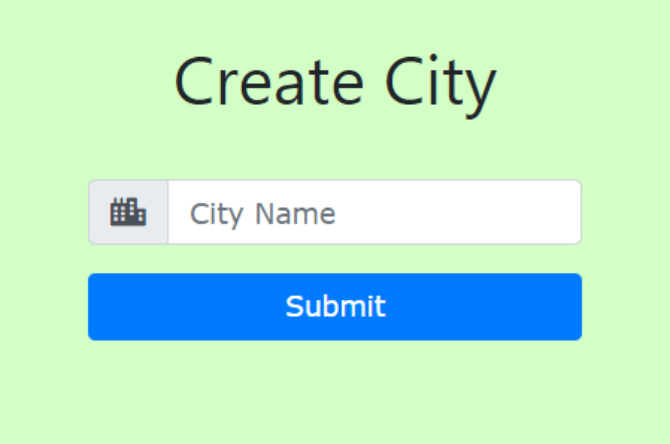
Εικόνα 47: Η σελίδα των αποτελεσμάτων μιας αναζήτησης

Στην αρχή της σελίδας εμφανίζεται μήνυμα στο οποίο φαίνεται ποια ήταν η αναζήτηση που εκτέλεσε ο χρήστης για να οδηγηθεί εδώ και με βάση την αναζήτηση που έχει πραγματοποιήσει εμφανίζεται μία λίστα με τα places που πληρούν τα κριτήρια αυτά. Το όνομα του place που θα εμφανιστεί είναι και link που επιλέγοντας το ο χρήστης θα μεταβεί στη σελίδα view playce που θα εμφανίσει της πληροφορίες του αντίστοιχου place.

Create City

Πρόσβαση σε αυτή τη σελίδα έχουν μόνο οι Admins του συστήματος. Είναι απαραίτητη για τυχόν επεκτάσεις που μπορεί να πραγματοποιηθούν στο σύστημα πχ αν υπάρξει ενδιαφέρον από μία περιοχή στην οποία μέχρι και τώρα δεν υπήρχε κανένα προς ενοικίαση οίκημα ο Admin μπορεί να εισάγει την περιοχή αυτή στη βάση δεδομένων και έτσι να αποκτήσει καινούριος πελάτες το σύστημα. Θα μπορούσαν πάντα τα δεδομένα αυτά να προϋπάρχουν στη βάση με τη λογική ότι θα μπορούσαμε να αποθηκεύσουμε όλες τις πόλεις που υπάρχουν στον κόσμο σε αυτήν ωστόσο αυτό θα ήταν πέρα από χρονοβόρο και κακό για την ίδια την εφαρμογή καθώς οποιοδήποτε query που απαιτεί αναζήτηση στον πίνακα cities θα καθυστερούσε σημαντικά.

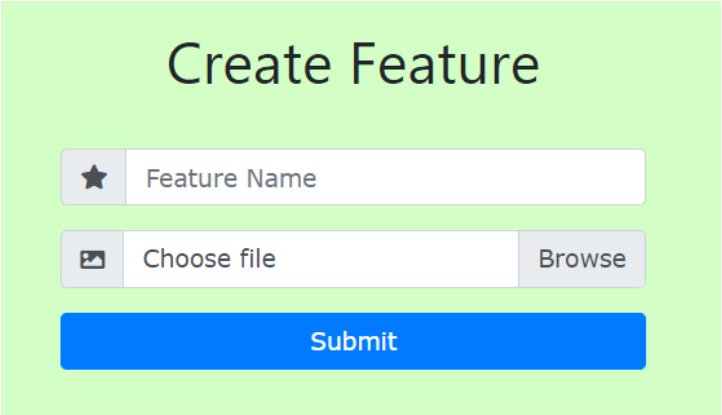
Η μορφή της σελίδας θα παρουσιαστεί παρακάτω. Ουσιαστικά είναι απλά μία σελίδα με ένα πεδίο text στο οποίο ο Admin εισάγει το όνομα της πόλης που θέλει να προσθέσει στο σύστημα και πατώντας το κουμπί submit πραγματοποιεί την εισαγωγή αυτή. Ο κώδικας με βάση τον οποίο εμφανίζεται η παρακάτω σελίδα αλλά έχει και λειτουργικότητα δεν κρίνεται απαραίτητο να παρουσιαστεί καθώς είναι ίδιας τεχνοτροπίας με τον κώδικα που παρουσιάστηκε για τη σελίδα Create Place.



Εικόνα 48: Η φόρμα δημιουργίας νέου city

Create Feature

Και σε αυτή τη σελίδα επιτρέπεται η πρόσβαση μόνο από όσους χρήστες έχουν τον ρόλο admin. Εδώ γίνεται η δημιουργία ενός feature προκειμένου να εμπλουτιστεί η βάση με νέες δυνατότητες που θα μπορούσε να προσφέρει κάποιος στο οίκημα του. Για παράδειγμα με την διάθεση στην αγορά μίας νέας τεχνολογίας συσκευής πλυντηρίου θα μπορούσε ένας χρήστης να θέλει να τονίσει ότι αν επιλέξει κάποιος το οίκημα του για ενοικίαση θα μπορούσε να βρει εκεί τη συσκευή αυτή και έτσι να προσελκύσει κόσμο. Η φόρμα έχει τη μορφή:



Εικόνα 49: Η φόρμα δημιουργίας νέου feature

Αποτελείται από δύο πεδία text και το κουμπί submit με το πάτημα του οποίου εκτελούνται όλες οι απαραίτητες εντολές για να δημιουργηθεί μία νέα εγγραφή στον πίνακα features. Το δεύτερο από τα πεδία text έχει την ιδιομορφία ότι είναι πεδίο επιλογής μοναδικής εικόνας στην οποία θα παρουσιάζεται το feature. Δίνεται ο κώδικας που χρειάζεται ούτως ώστε να μπορέσουν να γίνουν οι κατάλληλοι έλεγχοι για το τι αρχείο έχει επιλεγεί και αν αυτό υποστηρίζεται από τη βάση του συστήματος, αν υπάρχει ήδη το αρχείο στη βάση, αν το μέγεθος είναι κατάλληλο ενώ αν πληρεί όλες τις απαραίτητες προϋποθέσεις να αποθηκευθεί τελικά στη βάση αφού ανέβει στο σύστημα από το μέσο που χρησιμοποιεί ο admin.

```
75      // Image
76      $target_dir = "uploads/features/";
77      $target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
78      $uploadOk = 1;
79      $imageFileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
80      // Check if image file is a actual image or fake image
81      if (isset($_POST["submit"])) {
82          $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
83          if ($check !== false) {
84              echo "File is an image - " . $check["mime"] . ".";
85              $uploadOk = 1;
86          } else {
87              echo "File is not an image.";
88              $uploadOk = 0;
89          }
90      }
91      // Check if file already exists
92      if (file_exists($target_file)) {
93          echo "Sorry, file already exists.";
94          $uploadOk = 0;
95      }
96      // Check file size
97      if ($_FILES["fileToUpload"]["size"] > 500000) {
98          echo "Sorry, your file is too large.";
99          $uploadOk = 0;
100     }
101     // Allow certain file formats
102     if ($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg") {
103         echo "Sorry, only JPG, JPEG & PNG files are allowed.";
104         $uploadOk = 0;
105     }
106     // Check if $uploadOk is set to 0 by an error
107     if ($uploadOk == 0) {
108         echo "Sorry, your file was not uploaded.";
109         // if everything is ok, try to upload file
110     } else {
111         if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
112             $filename = $target_dir . basename($_FILES["fileToUpload"]["name"]);
113             $result = mysqli_query($link, "INSERT INTO features (name, image) VALUES ('$feature', '$filename')");
114             //echo "The file " . basename($_FILES["fileToUpload"]["name"]) . " has been uploaded.";
115         } else {
116             echo "Sorry, there was an error uploading your file.";
117         }
118     }
119     // End Image
```
Εικόνα 50: Ο κώδικας υλοποίησης της λειτουργίας ανάρτησης μιας εικόνας

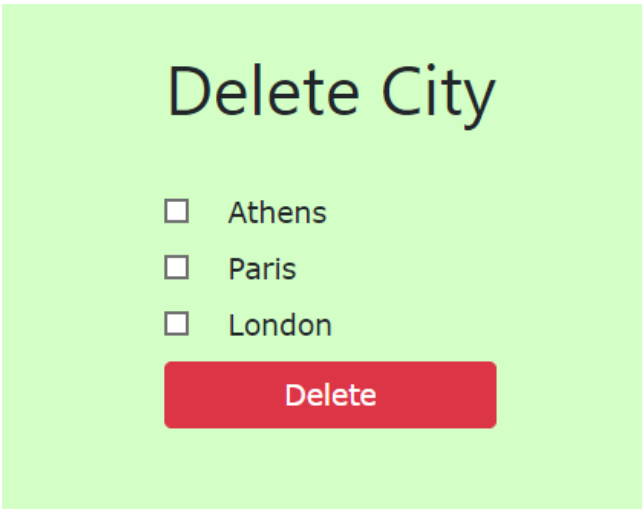
Approve Place

Εικόνα 51: Η σελίδα Approve Place

Η συγκεκριμένη σελίδα εξυπηρετεί τη λειτουργία Approve Place. Στη σελίδα αυτή εμφανίζονται όλα τα places στα οποία δεν είναι ακόμα ορατά στους χρήστες της εφαρμογής πέραν των admins. Ο admin βλέπει εδώ μία λίστα με όλα τα places που δεν έχουν ακόμα εγκριθεί και επιλέγοντας ένα place μεταβαίνει στη σελίδα View Place.

Delete City

Και αυτή η σελίδα αποτελεί μέρος των σελίδων που μπορεί κάποιος να δει μόνο αν έχει τον ρόλο Admin του συστήματος. Εδώ ο Admin μπορεί να διαγράψει μία πόλη όταν πλέον δεν χρειάζεται να υπάρχει στο σύστημα. Αυτό γίνεται με τη χρήση checkbox όπου ο admin μπορεί να τσεκάρει στο αντίστοιχο κουτάκι την πόλη ή τις πόλεις που θέλει να διαγράψει από τη βάση του συστήματος και να εκτελέσει την λειτουργία αυτή πατώντας το κουμπί Delete. Η σελίδα είναι ως εξής:



Εικόνα 52: Η σελίδα Delete City

```
<div class="card">
    <article class="card-body mx-auto" style="max-width: 400px;">
        <h1 class="card-title mt-3 text-center">Delete City</h1><br>

        <form name="city_form" method="post" action="">

            <?php
            require_once "config.php";

            // Get all cities as multiple checkbox select
            $result1 = mysqli_query($link, "SELECT * FROM cities");
            while ($row = $result1->fetch_assoc()) {
                unset($id, $name);
                $id = $row['c_id'];
                $name = $row['name'];
                echo '<div class="checkbox">
                        <label><input style="margin-right: 20px;" type="checkbox" name="city[]" value="' . $id . '">' . $name . '</label>
                    </div>';
            }
            ?>

            <div class="form-group">
                <button type="submit" name="sbmt" class="btn btn-danger btn-block"> Delete </button>
            </div>
        </form>
    </article>
</div>
```

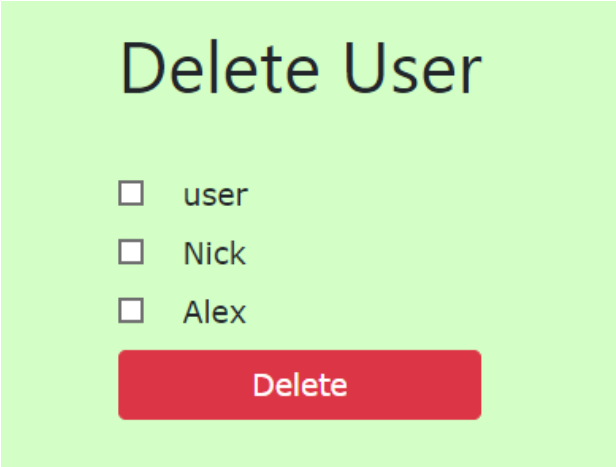
Εικόνα 53: Ο κώδικας δημιουργίας της σελίδας Delete City

```
if (isset($_POST['sbmt'])) {

    extract($_POST);

    if ($city) {

        $sql = "DELETE FROM cities WHERE c_id in ";
        $sql .= "('" . implode("','", array_values($city)) . "')";

        $result = mysqli_query($link, $sql);

        if (!$result) printf("Error during deletion.\n");
        else {
            Header('Location: '.$_SERVER['PHP_SELF']);
            mysqli_close($link);
        }
    }
}
```

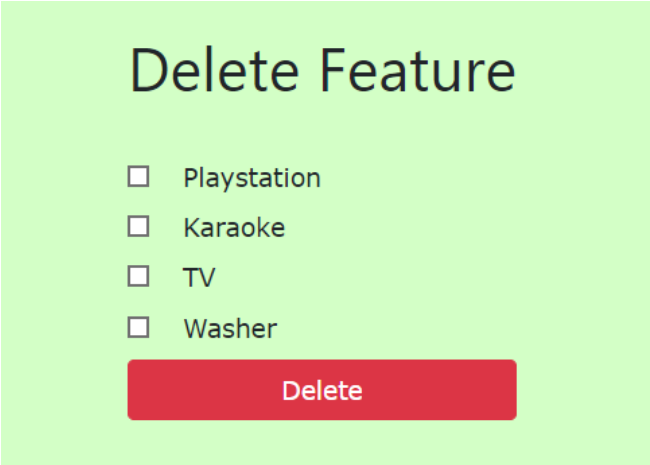
Εικόνα 54: Ο κώδικας υλοποίησης της λειτουργίας Delete City

Delete User – Delete Feature

Αντίστοιχα με τη σελίδα Delete City λειτουργούν και οι σελίδες Delete User και Delete Feature οπότε παρουσιάζεται μόνο η εμφάνιση της σελίδας όπως αυτή θα τη δει ο admin.



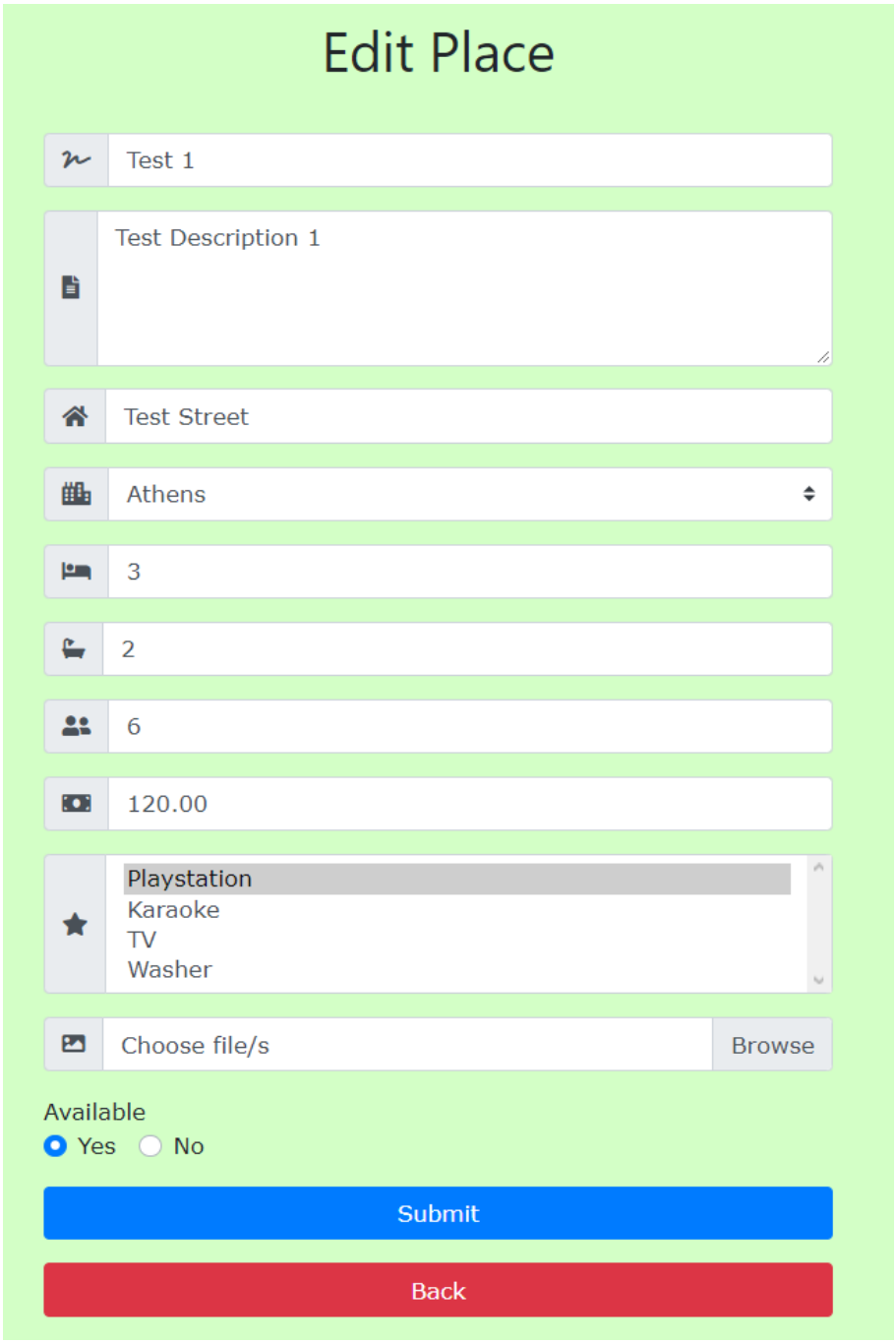
Εικόνα 55: Η σελίδα Delete User



Εικόνα 56: Η σελίδα Delete Feature

Edit Place

Η σελίδα Edit Place είναι ίδιας δομής με αυτή της create place ωστόσο εδώ υπάρχει ήδη προεπιλεγμένη τιμή για την πόλη καθώς ένα place δεν μπορεί να αλλάξει πόλη στην οποία βρίσκεται ενώ και η λειτουργία της διαφέρει στο γεγονός ότι δεν αποθηκεύει μία νέα εγγραφή στη βάση δεδομένων αντ' αυτού ενημερώνει τη βάση αλλάζοντας κάποια στοιχεία σε μία εγγραφή που υπάρχει ήδη.



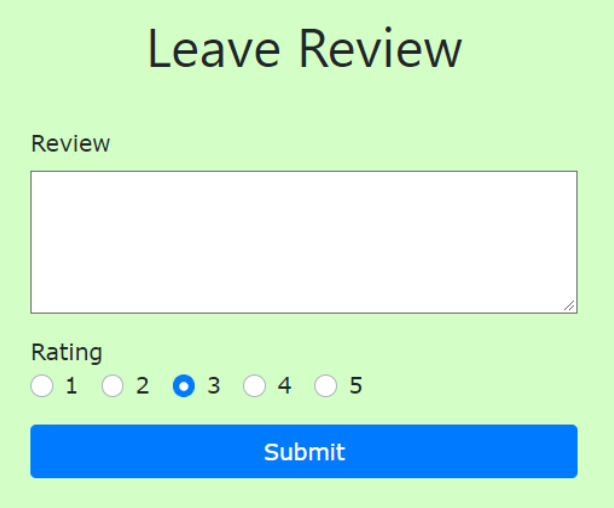
Εικόνα 57: Η φόρμα Edit Place

```
63              <!-- Title -->
64              <div class="form-group input-group">
65                  <div class="input-group-prepend">
66                      <span class="input-group-text"> <i class="fas fa-signature"></i> </span>
67                  </div>
68                  <input name="title" class="form-control" placeholder="Title" type="text" value="<?php echo $title; ?>">
69              </div>
70
```

Εικόνα 58: Ο κώδικας εμφάνισης ήδη υπαρχόντων δεδομένων κατά τη δημιουργία της φόρμας

Leave Review

Στη σελίδα αυτή μπορεί κάποιος να αφήσει τα σχόλια που έχει για την εμπειρία που έζησε αφού επισκέφτηκε ένα place, καθώς και τη βαθμολογία που δίνει στο μέρος αυτό με άριστα το πέντε ούτως ώστε να βγει ένας μέσος όρος για το place που θα βοηθήσει άλλους χρήστες να διαλέξουν ανάμεσα στα διάφορα places μίας περιοχής.



Εικόνα 59: Η φόρμα Leave Review

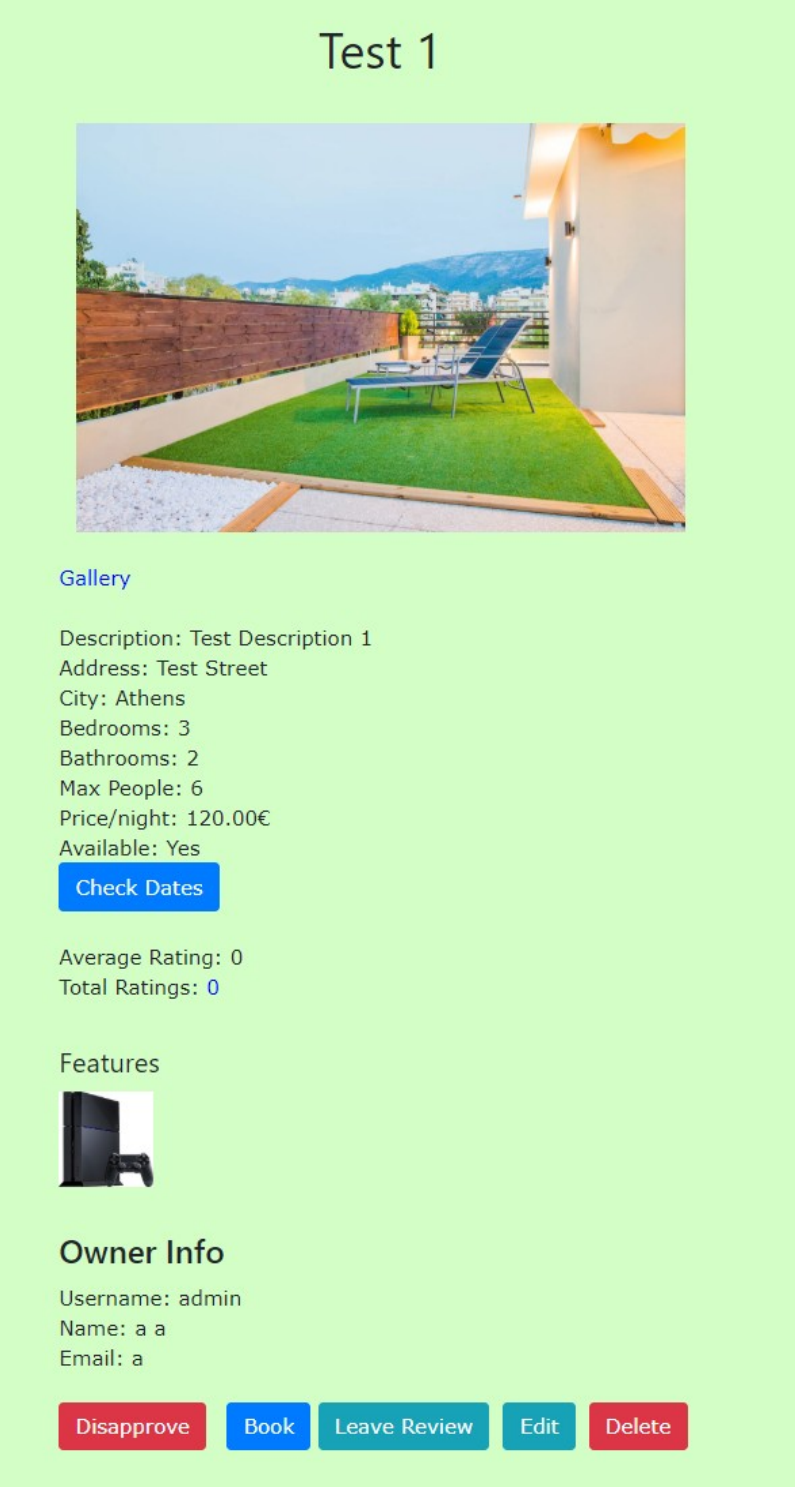
```
<!-- Rating -->
Rating<br>
<div class="form-group input-group">

    <div class="custom-control custom-radio custom-control-inline">
        <input type="radio" class="custom-control-input" value="1" id="rev_val1" name="rev_val">
        <label class="custom-control-label" for="rev_val1">1</label>
    </div>
    <div class="custom-control custom-radio custom-control-inline">
        <input type="radio" class="custom-control-input" value="2" id="rev_val2" name="rev_val">
        <label class="custom-control-label" for="rev_val2">2</label>
    </div>
    <div class="custom-control custom-radio custom-control-inline">
        <input type="radio" class="custom-control-input" value="3" id="rev_val3" name="rev_val" checked>
        <label class="custom-control-label" for="rev_val3">3</label>
    </div>
    <div class="custom-control custom-radio custom-control-inline">
        <input type="radio" class="custom-control-input" value="4" id="rev_val4" name="rev_val">
        <label class="custom-control-label" for="rev_val4">4</label>
    </div>
    <div class="custom-control custom-radio custom-control-inline">
        <input type="radio" class="custom-control-input" value="5" id="rev_val5" name="rev_val">
        <label class="custom-control-label" for="rev_val5">5</label>
    </div>
</div>
```

Εικόνα 60: Ο κώδικας δημιουργίας των radio buttons της φόρμας Leave Review

Αξίζει να σημειωθεί η διαφορετική έως τώρα τεχνική που χρησιμοποιήθηκε για την εισαγωγή των ratings. Το rating έχει τη μορφή radio button με προεπιλεγμένη την τιμή τρία. αυτό γίνεται ούτως ώστε ο χρήστης να μπορεί να επιλέξει μόνο μία τιμή με την οποία βαθμολογεί το place ενώ ταυτόχρονα βλέπει όλες τις επιλογές που μπορεί να έχει.

View Place

Στη σελίδα αυτή παρουσιάζονται όλες οι πληροφορίες που υπάρχουν για ένα συγκεκριμένο place.



Εικόνα 61: Η σελίδα View Place

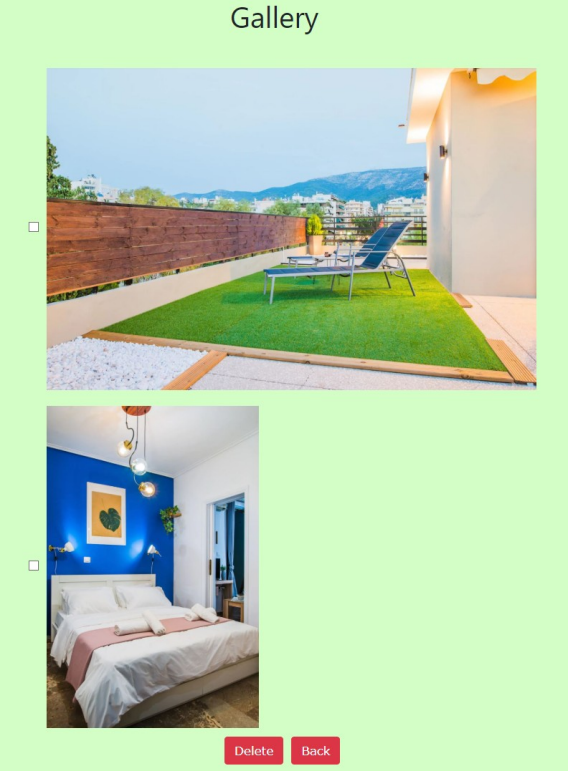
Οι επιλογές που εμφανίζονται στο χρήστη εξαρτώνται τόσο από το ρόλο του όσο και από το εάν είναι ιδιοκτήτης ή όχι. Αν ο ιδιοκτήτης του χώρου βλέπει τη σελίδα τότε του εμφανίζονται και οι επιλογές Edit ενώ αν στη σελίδα είναι οι διαχειριστές του συστήματος τότε εκτός των επιλογών book και leave review εμφανίζονται σε αυτούς και οι επιλογές Disapprove και Delete.

```php
<?php
require_once "config.php";
$result1 = mysqli_query($link, "SELECT * FROM users WHERE role = (SELECT r_id FROM roles WHERE name = 'user')");
while ($row = $result1->fetch_assoc()) {
    unset($id, $user);
    $id = $row['u_id'];
    $user = $row['username'];
    echo '<div class="checkbox">
            <label><input style="margin-right: 20px;" type="checkbox" name="users[]" value="' . $id . '">' . $user . '</label>
        </div>';
}
?>
```

Εικόνα 62: Ο κώδικας ανάκτησης συγκεκριμένου place

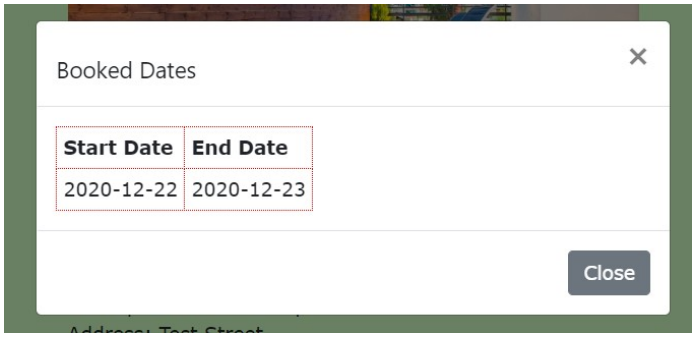
Σύμφωνα με τον κώδικα αυτόν γίνεται έλεγχος για τον ρόλο του User.

Στη σελίδα φαίνεται μία εκ των φωτογραφιών που έχει επιλέξει ο ιδιοκτήτης ενώ για να μπορέσει κάποιος να δει τις επιπλέον φωτογραφίες που μπορεί να υπάρχουν θα πρέπει να επιλέξει το Gallery που στην ουσία είναι ένα link και σε μεταφέρει στη σελίδα που εμφανίζονται όλες οι φωτογραφίες που υπάρχουν για το συγκεκριμένο place. Εκεί και πάλι με βάση τον ρόλο και εφόσον στη σελίδα είναι ο ιδιοκτήτης του χώρου δίνεται και η επιλογή για διαγραφή κάποιων φωτογραφιών.



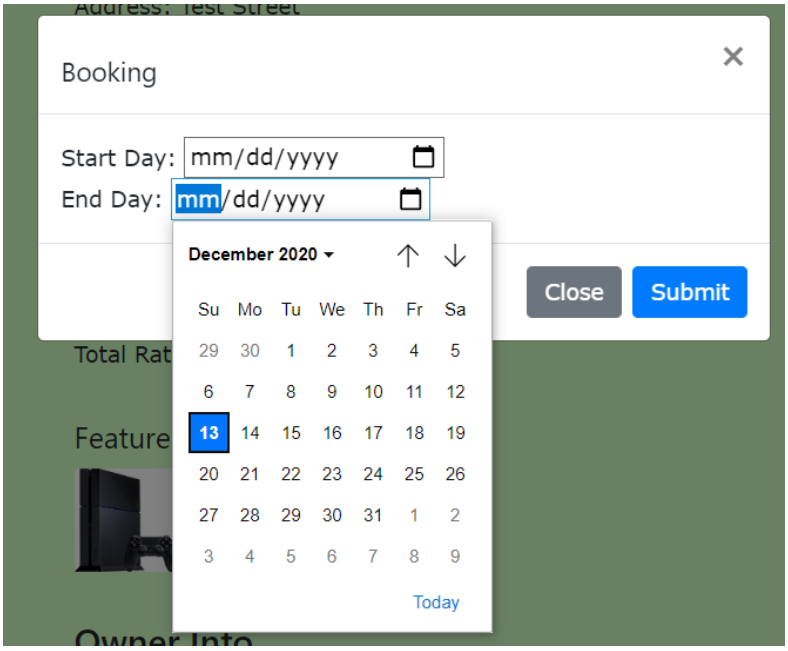
Εικόνα 63: Η σελίδα Gallery

Μετά το gallery ο χρήστης μπορεί να δει όλες της πληροφορίες του χώρου ενώ στο τέλος μπορεί να τσεκάρει τη διαθεσιμότητα του βλέποντας ποιες μέρες είναι δεσμευμένο από κρατήσεις άλλων χρηστών.



Εικόνα 64: Το παράθυρο Booked Dates

Για να πραγματοποιηθεί κράτηση πρέπει να γίνει επιλογή του Book. Πατώντας στο Book ο χρήστης βλέπει στην οθόνη ένα νέο παράθυρο στο οποίο του δίνεται η δυνατότητα να επιλέξει ημερομηνίες για τις οποίες θέλει να πραγματοποιήσει την κράτηση και με την επιλογή submit να την υποβάλει.



Εικόνα 65: Το παράθυρο υλοποίησης του booking

Ο κώδικας που πραγματοποιεί την κράτηση είναι ο εξής:

```php
if (isset($_POST['sbmt'])) {

    extract($_POST);

    // Check if not null
    if ($start && $end) {
        // Check if valid dates
        if (strcmp($start, $end) <= 0 && strcmp($start, $date_now) >= 0 && strcmp($end, $date_now) >= 0) {
            // Check if free dates
            $result = mysqli_query($link, "SELECT * FROM bookings WHERE place = '$id'");
            while ($row = $result->fetch_assoc()) {
                unset($start_date, $end_date);
                $start_date = $row['start_date'];
                $end_date = $row['end_date'];

                if (!(strcmp($end, $start_date) < 0 || strcmp($start, $end_date) > 0)) {
                    $is_free = 0;
                    break;
                }
            }

            // Insert booking
            if ($is_free == 1) {
                $sql = "INSERT INTO bookings (place, user, start_date, end_date) VALUES ('$id', '$logged_user', '$start', '$end')";
                $result2 = mysqli_query($link, $sql);
            }
```

Εικόνα 66: Ο κώδικας υλοποίησης της λειτουργίας booking

Τέλος οι χρήστες μπορούν να δουν το rating του χώρου, μία πληροφορία που τους δείχνει πόσο ευχαριστημένοι ή όχι είναι προηγούμενοι επισκέπτες του χώρου. Το rating βγαίνει σε ποσοστό και ο βαθμός του χώρου μπορεί να είναι από μηδέν έως πέντε.

```php
    $result3 = mysqli_query($link, "SELECT count(rating), round(AVG(rating),1) from reviews where place='$id'");
    while ($row3 = $result3->fetch_assoc()) {
        unset($avg, $count);
        $avg = $row3['round(AVG(rating),1)'];
        $count = $row3['count(rating)'];
    }
```

Εικόνα 67: Ο κώδικας εμφάνισης των ratings ενός place

6.2 Wordpress

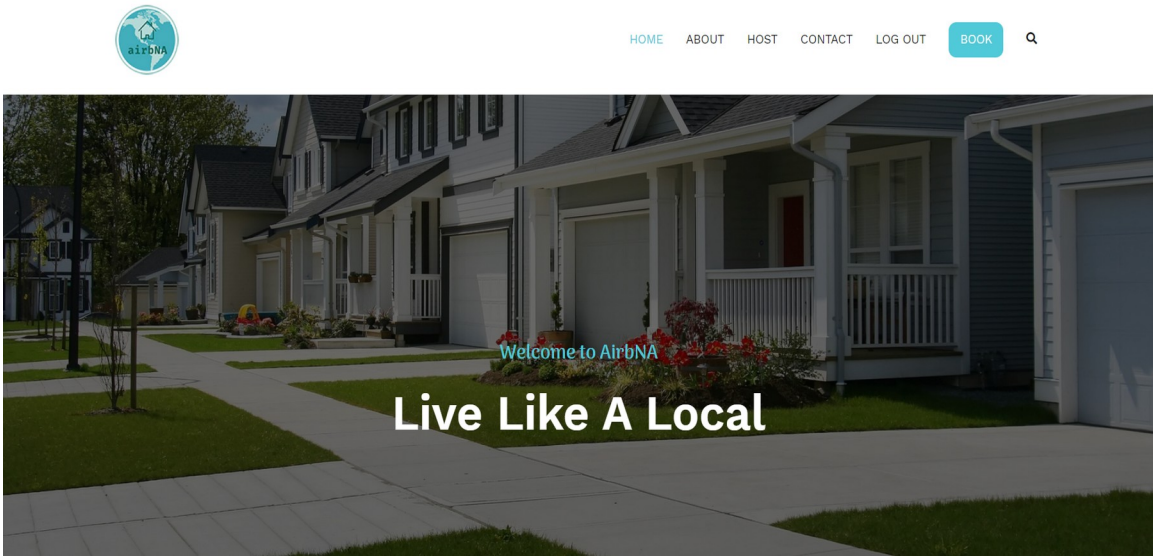
Όπως έχει αναφερθεί το wordpress είναι ένα σύστημα διαχείρισης περιεχομένου (Content Management System - CMS). Για να στηθεί η εφαρμογή δεν χρειάζονται ιδιαίτερες γνώσεις προγραμματισμού από τον προγραμματιστή καθώς όλα γίνονται με τη χρήση themes και plugins.

Για τη δημιουργία της συγκεκριμένης εφαρμογής χρησιμοποιήθηκαν τα εξής plugins:

1. **BAW Login/Logout Menu**: με αυτό το plugin προστίθεται στην εφαρμογή η λειτουργία login/logout. Το ίδιο το plugin έχει τη δυνατότητα να αναγνωρίζει πότε ένας χρήστης είναι συνδεδεμένος και πότε όχι και έτσι να εναλλάξει αν ο χρήστης θα βλέπει την επιλογή login ή logout αντίστοιχα.

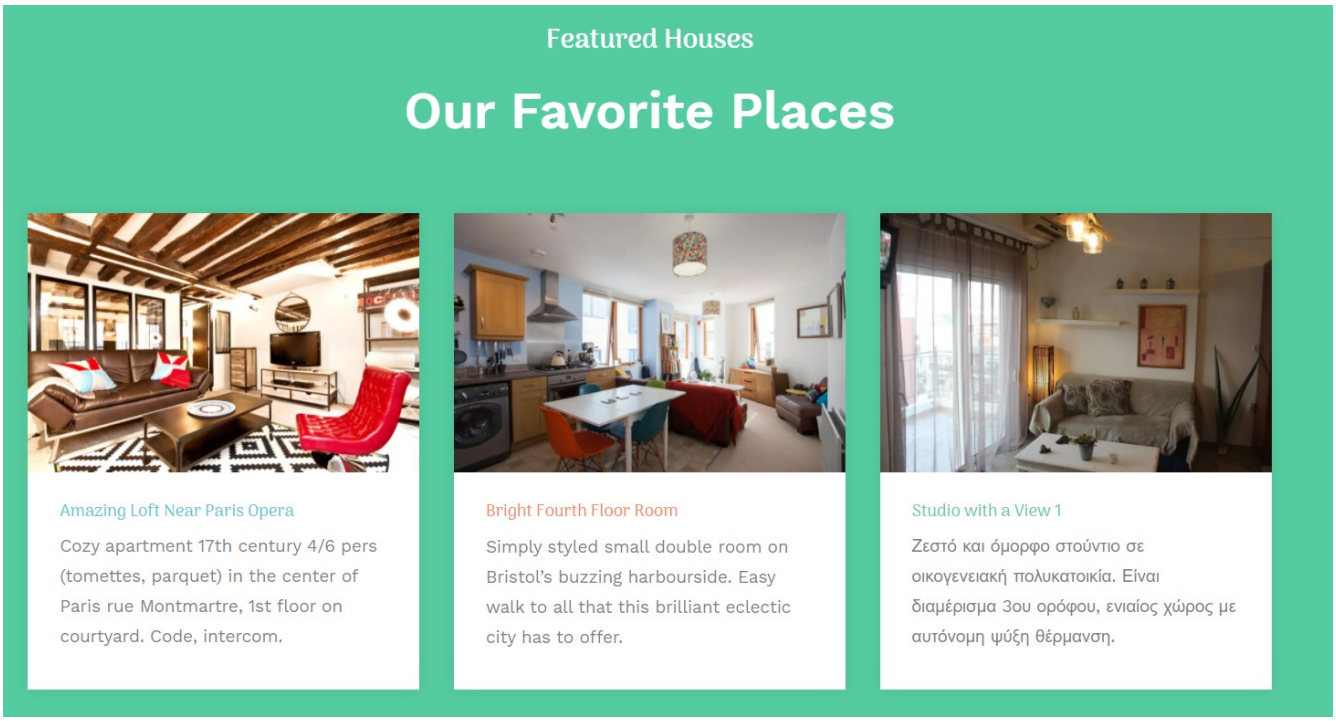
2. **Duplicate Page**: επιτρέπει την δημιουργία duplicate σελίδων οπότε έγινε χρήση για εξοικονόμηση χρόνου κατά τη διάρκεια κατασκευής της εφαρμογής.
3. **Elementor**: είναι ένα plugin που χρησιμοποιείται για να φτιάξει ο προγραμματιστής το design της σελίδας. Προσφέρει άμεση αλληλεπίδραση με το site και μπορεί σε ζωντανό χρόνο να δει τις αλλαγές που έχει πραγματοποιήσει.
4. **Everest Forms**: παρέχει τη δυνατότητα δημιουργίας οποιασδήποτε φόρμας με εύκολο τρόπο φιλικό προς τον προγραμματιστή.
5. **Hotel Booking Lite**: παρέχει ένα ολόκληρο σύστημα booking για δωμάτια ξενοδοχείων. Σημειώνεται ότι για τις ανάγκες της εφαρμογής έγινε τροποποίηση του plugin και αντί για δωμάτια ξενοδοχείων αποθηκεύονται και εμφανίζονται οι χώροι ενοικίασης.
6. **Memphis Custom Login**: προσφέρει έναν εύκολο τρόπο να μορφοποιηθεί η σελίδα με το login. Δίνει επιλογές σε αλλαγή τόσο στα χρώματα όσο και στα logo της σελίδας κάνοντας την πιο προσωπική για τις ανάγκες του προγραμματιστή.
7. **Say What?**: επιτρέπει την αλλαγή στο κείμενο της σελίδας χωρίς να χρειαστεί τροποποίηση μέσω του κώδικα ενός plugin ή του ίδιου του πυρήνα του wordpress κώδικα. Αλλάζει το κομμάτι που επιλέγει ο προγραμματιστής με το νέο κείμενο που θα δηλώσει στο plugin.
8. **User Role Editor**: επιτρέπει την διαχείριση των ρόλων που υπάρχουν στην εφαρμογή. Είναι αρκετά απλό στη χρήση αφού αρκεί να τσεκάρει ο admin τις νέες δυνατότητες που θέλει να δώσει στον ρόλο και να κάνει update το ρόλο αυτό για να αποθηκευτούν οι αλλαγές.

Front end



Εικόνα 68: Η αρχική σελίδας της εφαρμογής Wordpress

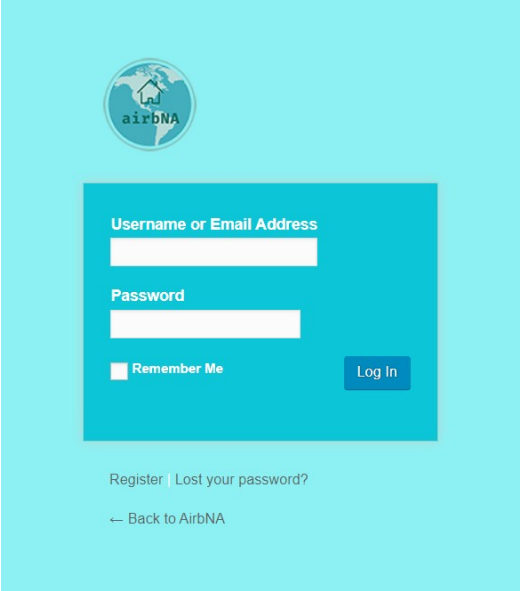
Η αρχική σελίδα που θα δει ο χρήστης που θα επισκεφτεί την εφαρμογή περιλαμβάνει τη μπάρα/μενού στο πάνω κομμάτι της σελίδας όπως βλέπουμε και στην παραπάνω εικόνα καθώς και ένα μήνυμα καλωσορίσματος στην εφαρμογή. Στη μπάρα υπάρχουν στοιχεία όπως είναι το logo της εφαρμογής και στη συνέχεια τα link/επιλογές. Το Home που σε επιστρέφει πάντα στην αρχική σελίδα, το About μεταφέρει το χρήστη στη σελίδα με τις πληροφορίες της εφαρμογής. Η επιλογή Host μεταφέρει το χρήστη στη σελίδα όπου μπορεί να πραγματοποιήσει την εισαγωγή του χώρου του στην εφαρμογή ούτως ώστε να μπορούν οι άλλοι χρήστες να τον επιλέξουν για ενοικίαση. Στο Contact ο χρήστης μπορεί να βρει πληροφορίες ούτως ώστε να επικοινωνήσει με τους υπεύθυνους της εφαρμογής ενώ τέλος στις επιλογές Login/Logout ο χρήστης πραγματοποιεί σύνδεση στην εφαρμογή και αντίστοιχα αποσύνδεση από αυτή. Οι επιλογές Book και Search (μεγεθυντικός φακός) όπως δηλώνει και το όνομα τους οδηγούν τον χρήστη στη σελίδα όπου μπορεί να πραγματοποιήσει τα bookings του είτε στη σελίδα των αποτελεσμάτων της αναζήτησης που πραγματοποίησε.



Εικόνα 69: Το τμημα "Featured Houses" της αρχικής σελίδας

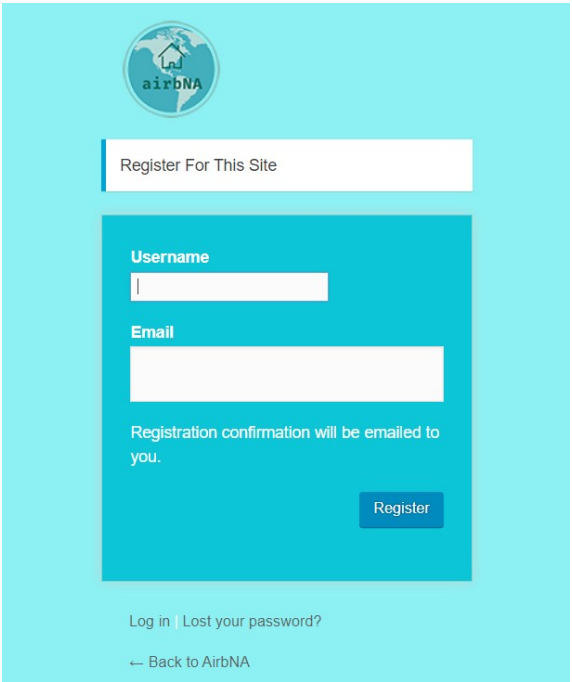
Στη συνέχεια της αρχικής σελίδας ο χρήστης μπορεί να δει και τους αγαπημένους χώρους ενοικίασης που έχουν επιλεχτεί από τον admin.

Login - Register



Εικόνα 70: Η σελίδα Login

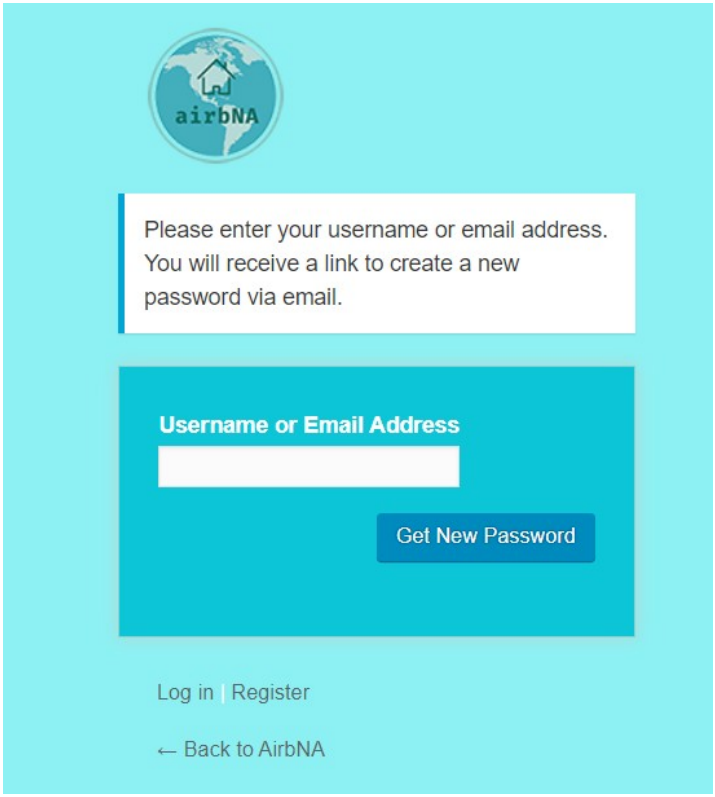
Επιλέγοντας ο χρήστης στην αρχική σελίδα την επιλογή Login μεταφέρεται σε αυτή τη σελίδα όπου χρειάζεται να εισάγει είτε το username του είτε το email του καθώς και τον κωδικό του. Αν ο χρήστης δεν έχει ήδη εγγραφεί στην εφαρμογή τότε μπορεί να επιλέξει την επιλογή Register που τον μεταφέρει στην παρακάτω σελίδα:



Εικόνα 71: Η σελίδα Register

όπου ο χρήστης εισάγει τα στοιχεία που του ζητούνται στα αντίστοιχα πεδία και έτσι πραγματοποιεί εγγραφή στην εφαρμογή.

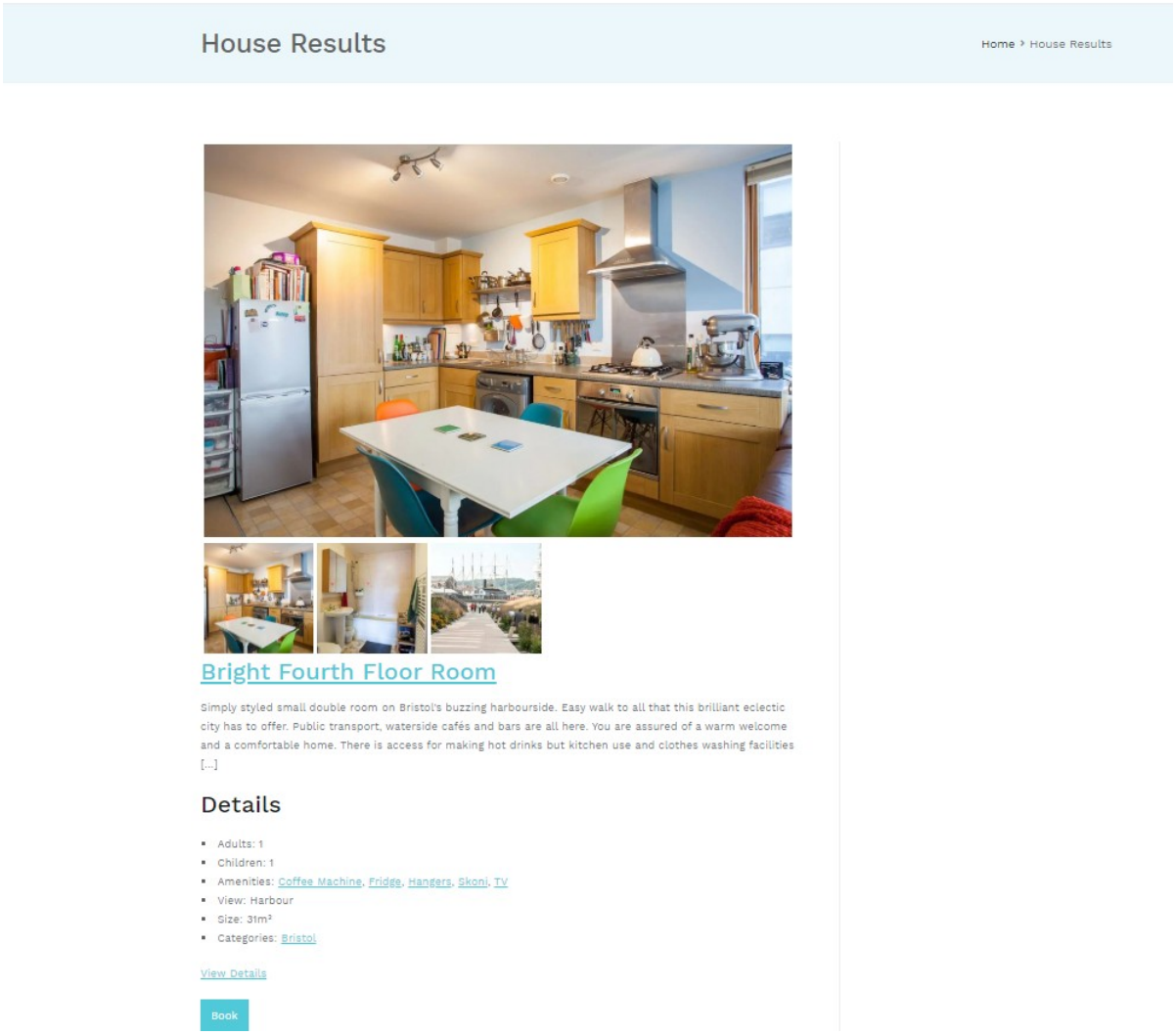
Αν κάποιος χρήστης έχει ξεχάσει τον κωδικό του τότε αρκεί να επιλέξει την επιλογή Lost your password και να μεταφερθεί σε μία άλλη σελίδα όπου χρειάζεται να εισάγει είτε το username είτε το email του και να του δοθεί ένας νέος κωδικός.



Εικόνα 72: Η σελίδα αλλαγής κωδικού

Search Results – Place Details

Είναι η σελίδα στην οποία θα μεταφερθεί ο χρήστης αφού πραγματοποιήσει ανίχνευση με βάση το όνομα ενός χώρου ή την πόλη στην οποία θέλει να βρει χώρους προς ενοικίαση. Σε αυτή τη σελίδα θα βρει όλους τους χώρους που ανταποκρίνονται στα δεδομένα που έχει δώσει.

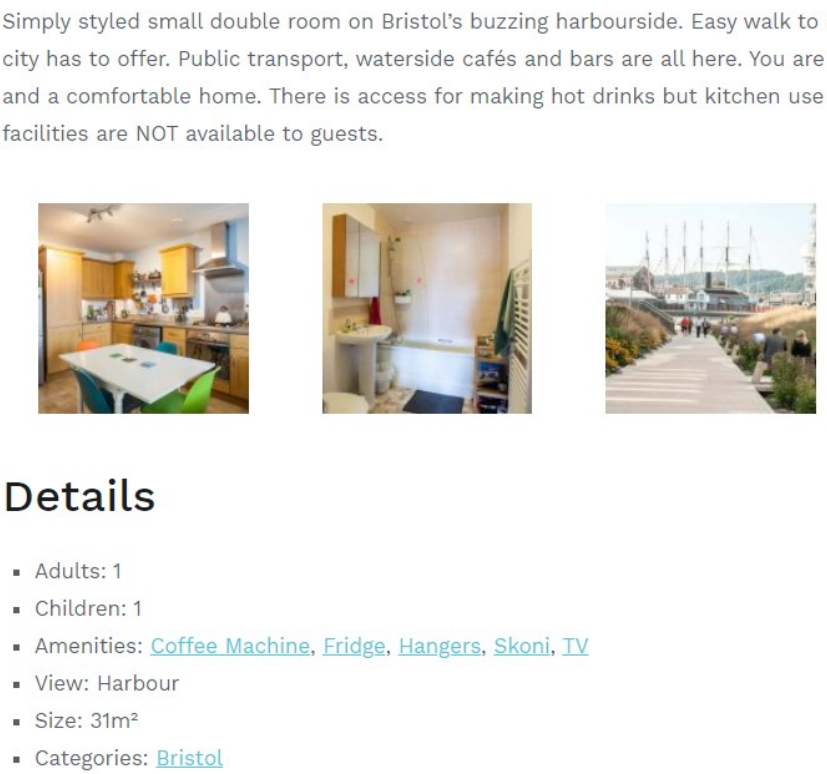


Εικόνα 73: Η σελίδα House Results

Ο χρήστης μπορεί να δει αναλυτικά όλες τις πληροφορίες του χώρου. Μέσα σε αυτή τη σελίδα του δίνονται επιπλέον επιλογές. Αν ο χρήστης επιλέξει κάποιο από τα feature ή το category τότε μεταφέρεται σε μία σελίδα όπου παίρνει σαν αποτέλεσμα κάθε χώρο που ανταποκρίνεται στα ίδια κριτήρια, θα δει δηλαδή κάθε χώρο που περιλαμβάνει το ίδιο feature που επέλεξε ή κάθε χώρο που υπάρχει στην ίδια πόλη. Η επιλογή View Details τον μεταφέρει στη σελίδα όπου μπορεί να δει περισσότερες πληροφορίες για τον συγκεκριμένο χώρο ενώ η επιλογή Book στη σελίδα που πραγματοποιείται το booking όπως αναφέρθηκε και προηγουμένως.

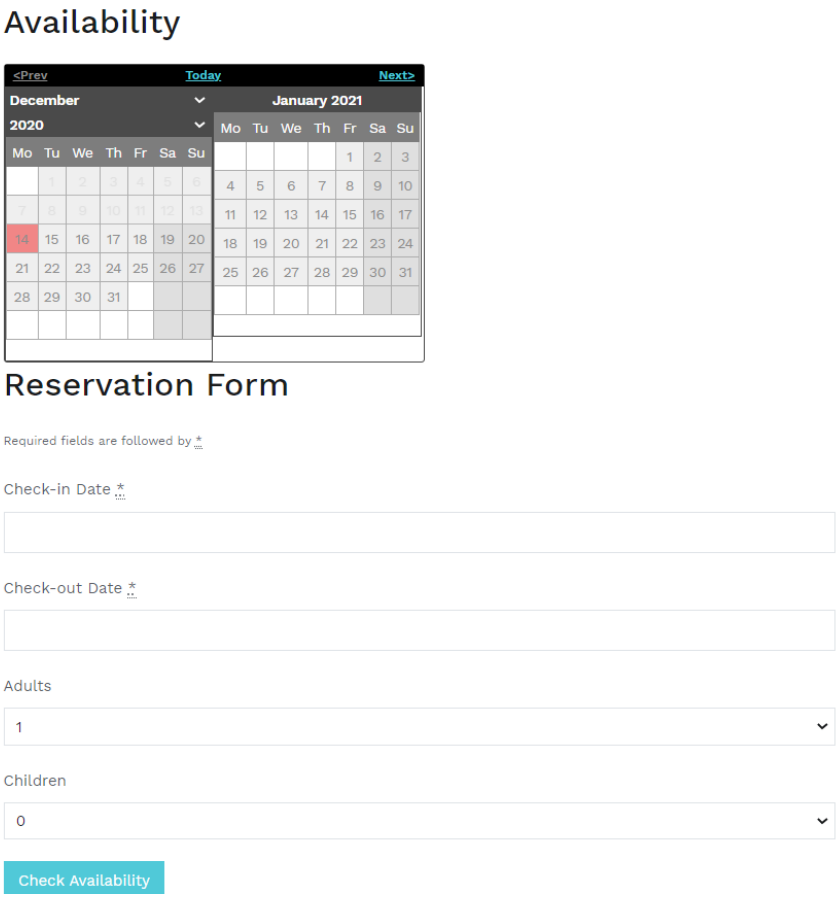
Επιλέγοντας τον τίτλο του αποτελέσματος ο χρήστης μεταφέρεται στη σελίδα με τις επιπλέον πληροφορίες του χώρου.

Εικόνα 74: Η σελίδα προβολής ενός οικήματος



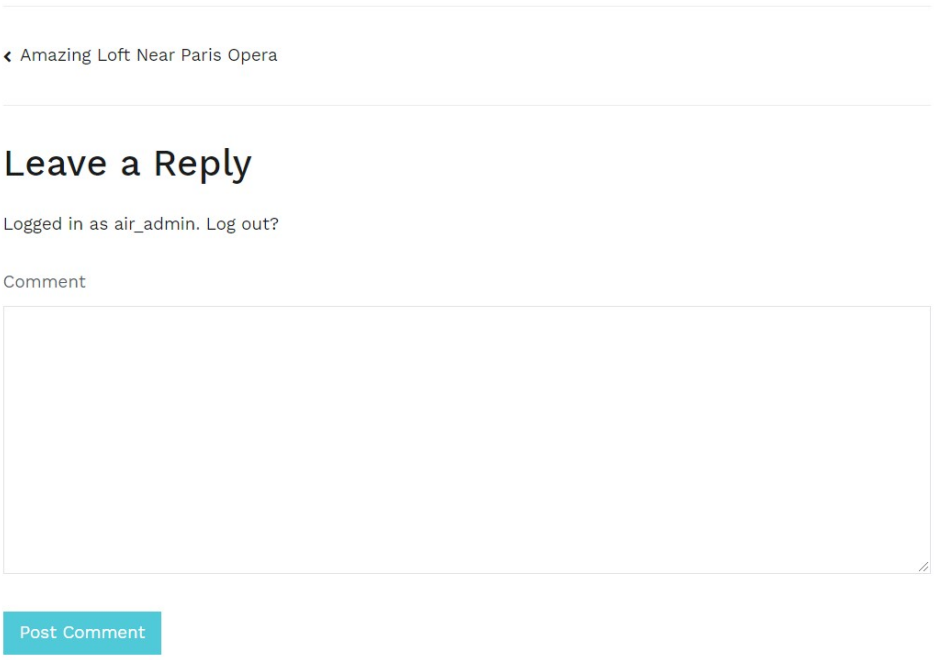
Εικόνα 75: Οι λεπτομέρειες και τα στοιχεία ενός οικήματος

Μετά την κεντρική εικόνα του χώρου ακολουθεί σύντομη περιγραφή του χώρου και οι επιπλέον φωτογραφίες. Έπειτα από τις αναλυτικές πληροφορίες του χώρου ο χρήστης μπορεί να ελέγξει την διαθεσιμότητα του. Αυτό γίνεται εισάγοντας ημερομηνίες άφιξης και αποχώρησης καθώς και πόσα άτομα έχουν σκοπό να επισκεφτούν το χώρο.



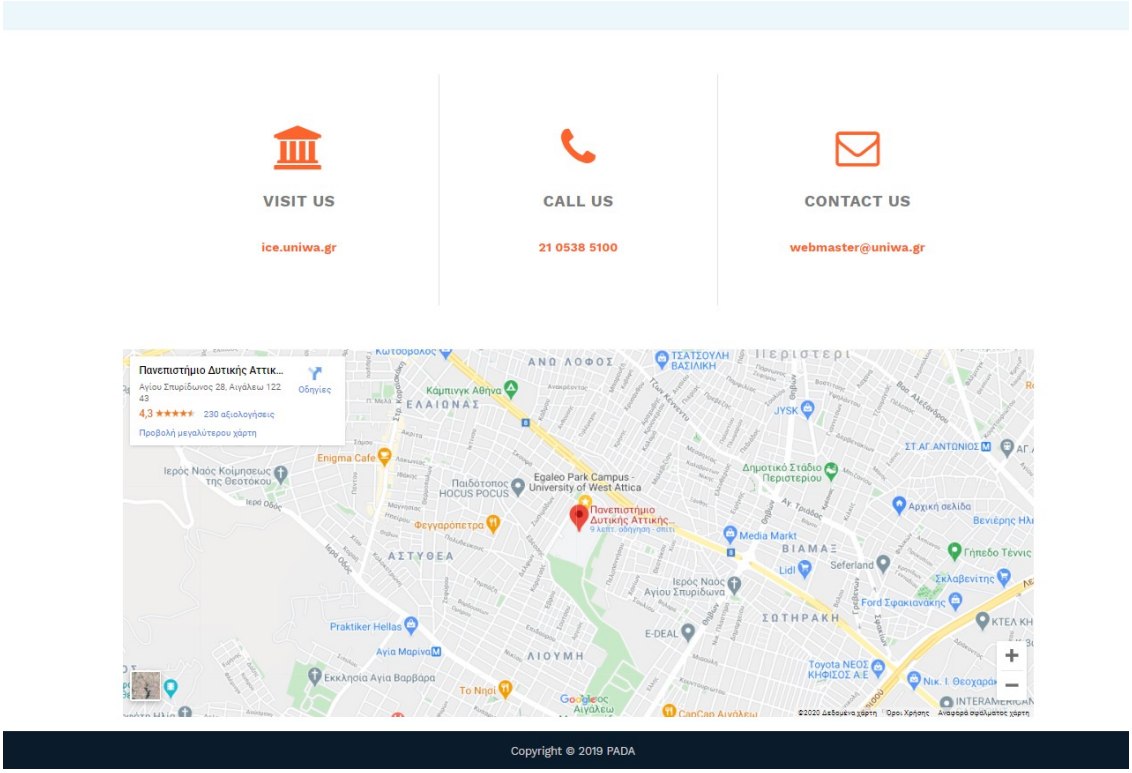
Εικόνα 76: Το τμήμα ελέγχου διαθεσιμότητας ενός οικήματος

Στη συνέχεια μπορεί να δει τις αξιολογήσεις και τα σχόλια που έχουν αφήσει ήδη άλλοι χρήστες ή να πραγματοποιήσει τα δικά του.



Εικόνα 77: Η φόρμα Leave a Comment

Contact

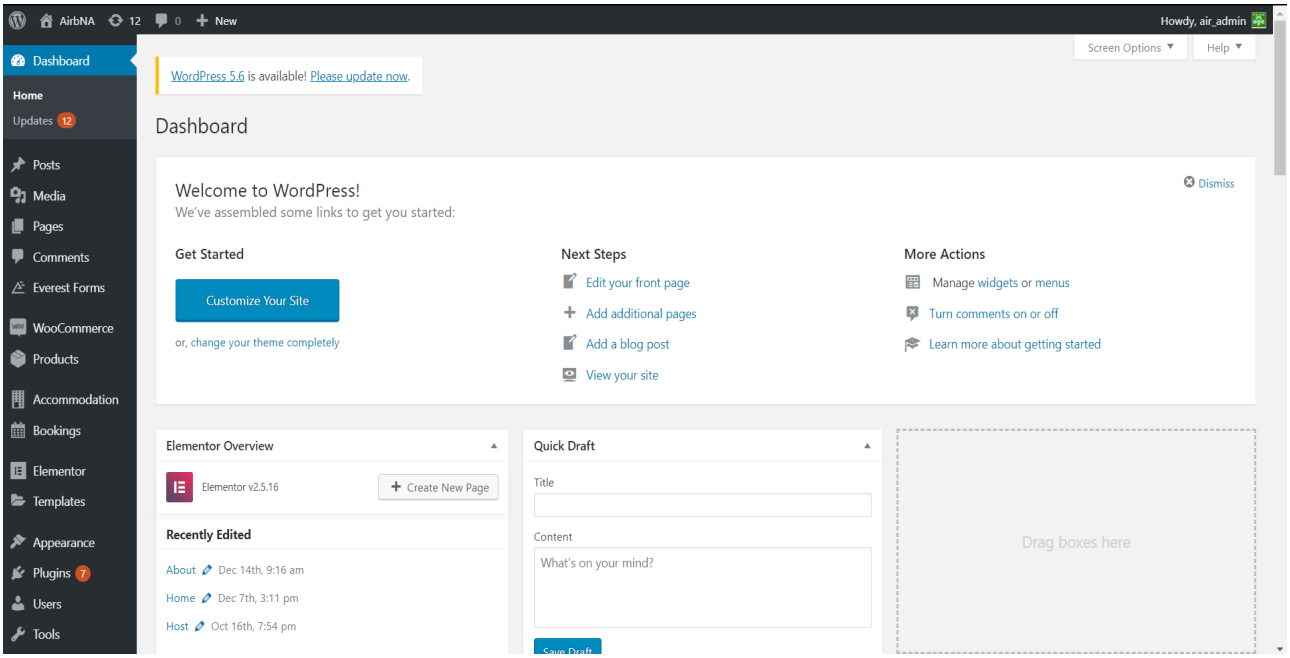


Εικόνα 78: Η σελίδα Contact

Όπως αναφέρθηκε στη σελίδα αυτή ο χρήστης μπορεί να βρει όλες τις πληροφορίες που χρειάζεται ώστε να επικοινωνήσει με τους admin της εφαρμογής.

Πρέπει να σημειωθεί ότι έγινε η εξής παραδοχή. Για να πραγματοποιηθεί η ενέργειά Host και ένας χρήστης να μπορέσει να θέσει το χώρο του προς ενοικίαση από την εφαρμογή είναι αναγκαία η επικοινωνία του με τον admin στον οποίο και δίνει όλα τα στοιχεία του χώρου του ούτως ώστε ο admin με τη σειρά του να τα εισάγει στην εφαρμογή. Η ενέργεια Host δηλαδή δεν πραγματοιείται κατευθείαν από τον χρήστη.

Back End

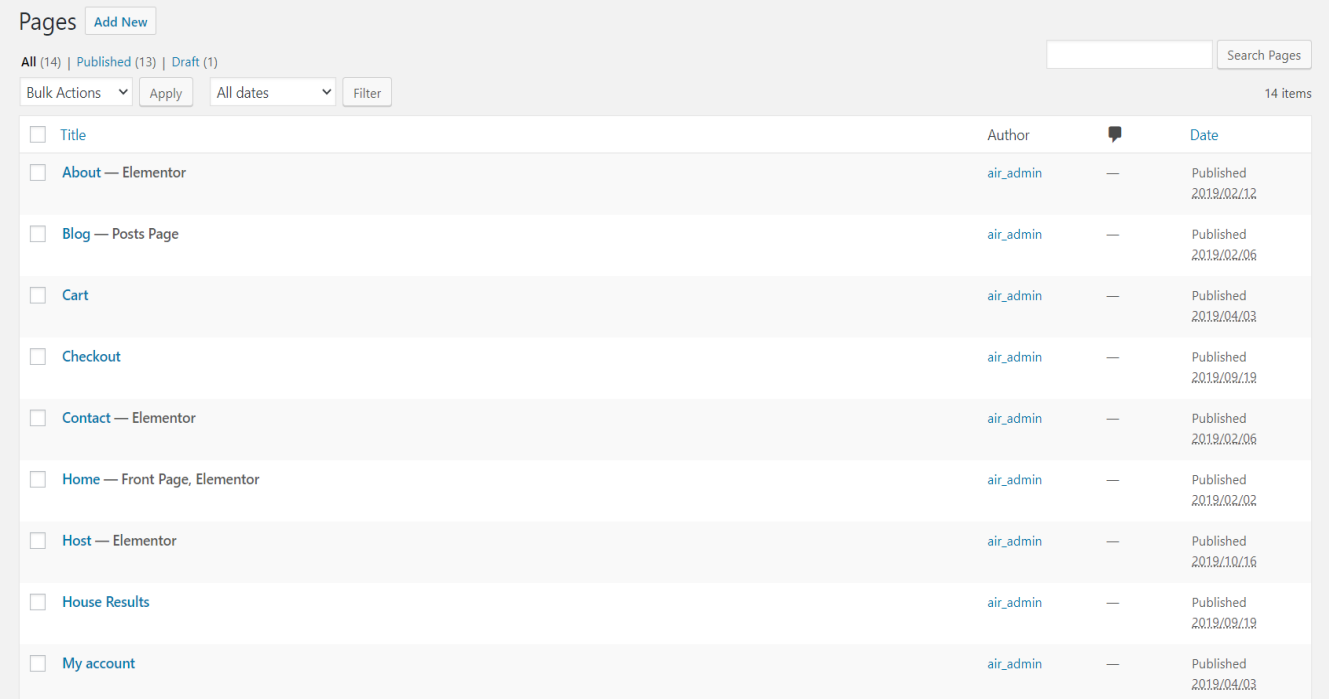


Εικόνα 79: Το Dashboard του admin

Αυτή είναι η αρχική σελίδα που θα φορτώσει στον Admin όταν μπει για να πραγματοποιήσει οποιαδήποτε αλλαγή στην εφαρμογή. Όπως φαίνεται στην αριστερή πλευρά υπάρχει ένα μενού επιλογών μέσα από τις οποίες μπορεί να πραγματοποιήσει τις αλλαγές που θέλει ο admin. Παραδείγματος χάρη αν θέλει να προσθέσει κάποιο plugin, ή να διαγράψει θα επιλέξει την επιλογή Plugins.

Pages

Αν ο χρήστης κατευθυνθεί στην επιλογή Pages τότε μπροστά του θα δει πληροφορίες για τις σελίδες που υπάρχουν στην εφαρμογή. Βλέπει πληροφορίες όπως ποιός χρήστης έχει επεξεργαστεί την σελίδα και πότε, ενώ μπορεί να επιλέξει διάφορες σελίδες που θέλει και να τις επεξεργαστεί είτε μία μία, είτε ταυτόχρονα.

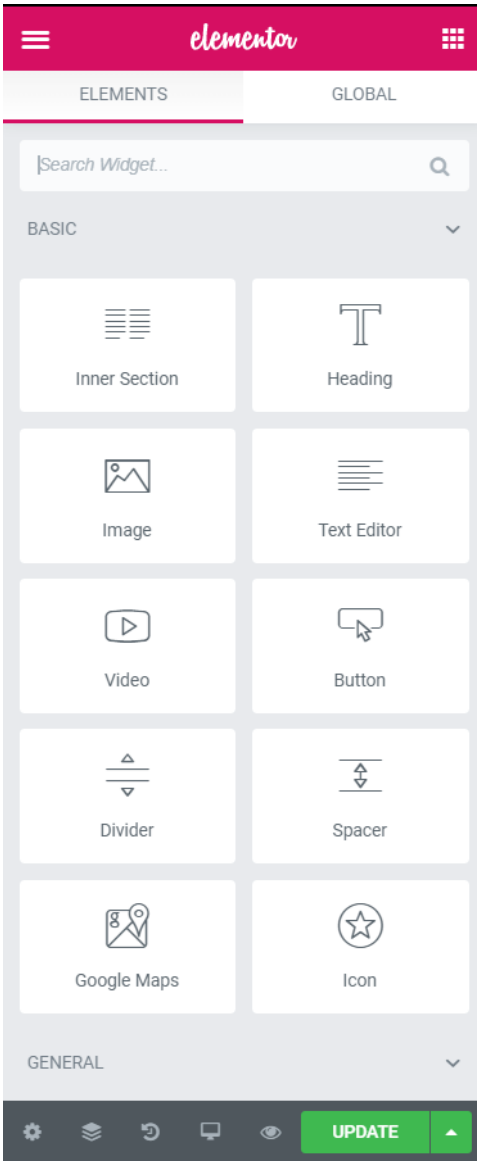


Εικόνα 80: Η σελίδα προβολής σελιδών της εφαρμογής

Edit Page

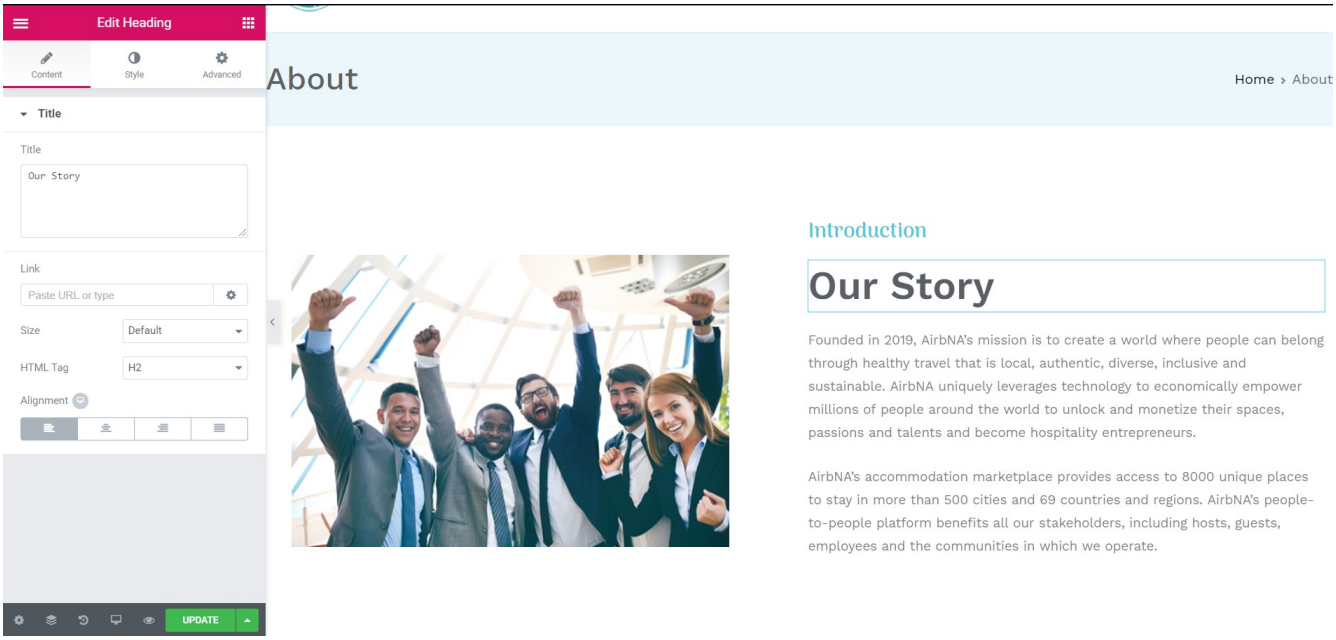
Εδώ οι Admins μπορούν να επεξεργαστούν οποιαδήποτε σελίδα ξεχωριστά. Μέσω του μενού επιλογών που τους δίνεται έχουν τη δυνατότητα να επεξεργαστούν την εσωτερική δομή της σελίδας (τίτλος, εικόνες, βίντεο, εικόνες) επιλέγοντας την επιλογή Elements και κατόπιν μία από τις βασικές επιλογές.

Το μενού που βλέπουν οι Admins είναι το εξής:



Εικόνα 81: Το αρχικό μενού επεξεργασίας μιας σελίδας

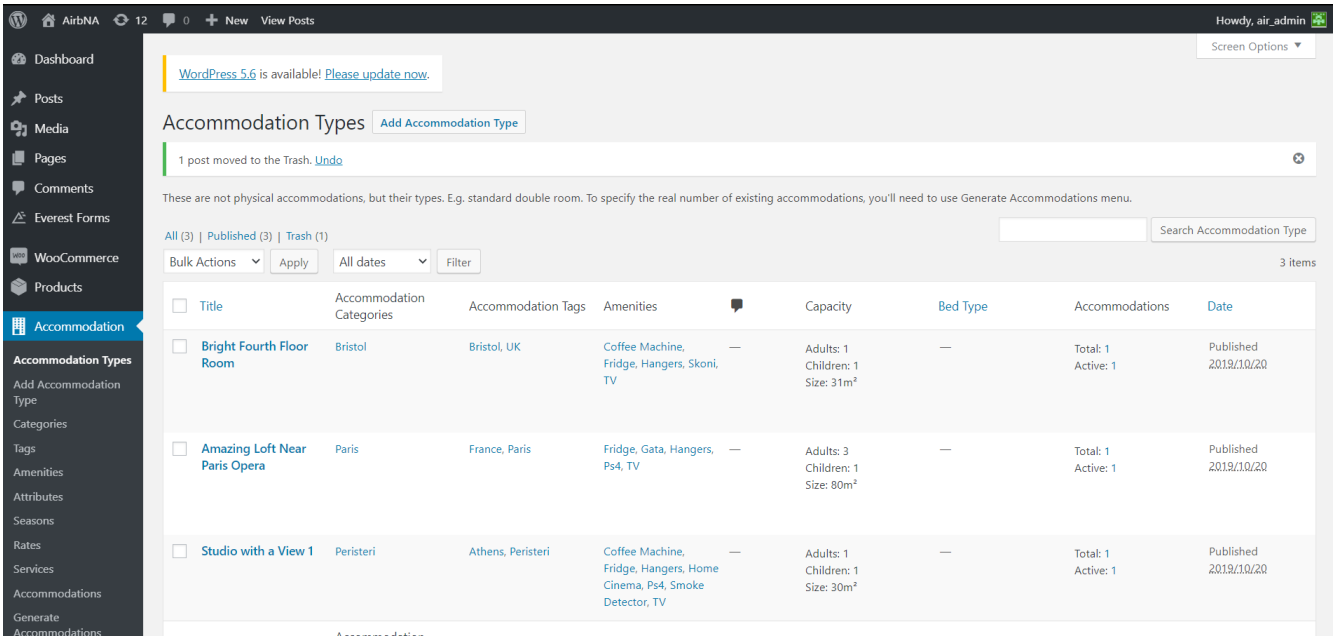
Όταν επιλεγεί η ξεχωριστή διαμόρφωση των χαρακτηριστικών της σελίδας τότε οι admins έχουν μπροστά τους και το μενού επιλογών αλλά και το πως φαίνεται η σελίδα χάρη στο Elementor plugin που χρησιμοποιείται. Όπως φαίνεται στην αριστερή πλευρά της σελίδας ο χρήστης βλέπει τις επιλογές που έχει για να κάνει edit την σελίδα, μπορεί να επεξεργαστεί τον τίτλο, το link, το style ενώ όποια αλλαγή και εάν εφαρμόσει μπορεί να την δει αυτόματα στο δεξί μέρος της σελίδας που βλέπει την ιστοσελίδα που επεξεργάζεται αυτή τη στιγμή. Με τον τρόπο αυτό έχει άμεση επαφή με τις αλλαγές που εφαρμόζει.



Εικόνα 82: Το μενού επεξεργασίας ενός στοιχείου μιας σελίδας

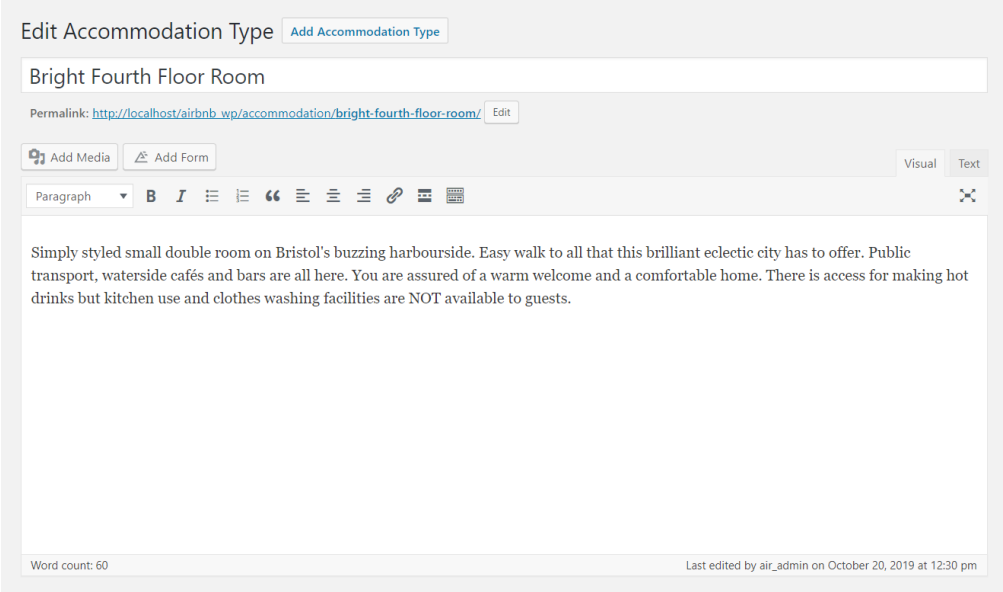
Accommodation

Μία βασική λειτουργία που δίνεται στους admin είναι αυτή των accommodation. Εδώ υπάρχει οτιδήποτε έχει να κάνει με τους χώρους ενοικίασης της εφαρμογής. Βασισμένο στο plugin Hotel Booking Lite παρέχει τη δυνατότητα επεξεργασίας του κάθε χώρου ξεχωριστά και την εισαγωγή όλων τον στοιχείων που έχουν να κάνουν με αυτόν όπως είναι η πόλη, τα amenities και διάφορα tags ή κατηγορίες.



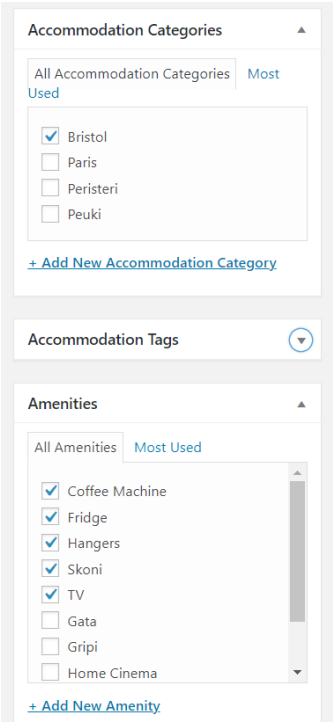
Εικόνα 83: Η σελίδα προβολής των οικημάτων

Επιλέγοντας έναν χώρο ο admin θα μεταφερθεί σε άλλη σελίδα με τις ειδικές επιλογές επεξεργασίας του χώρου αυτού.



Εικόνα 84: Η φόρμα επεξεργασίας των βασικών πληροφοριών ενός οικήματος

Εδώ καλείται να συμπληρώσει τις πληροφορίες για τον τίτλο και την περιγραφή του χώρου.

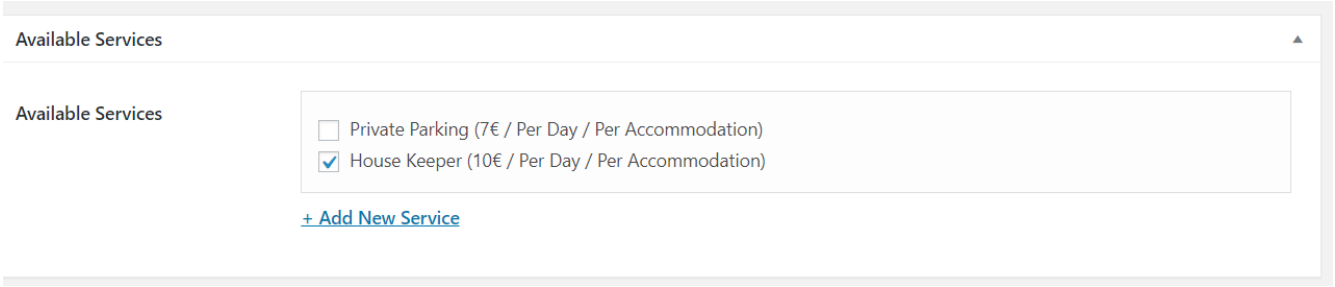


Εικόνα 85: Οι φόρμες αλλαγής της πόλης και των παροχών

Εικόνα 86: Η φόρμα αλλαγής του ορίου των επισκεπτών

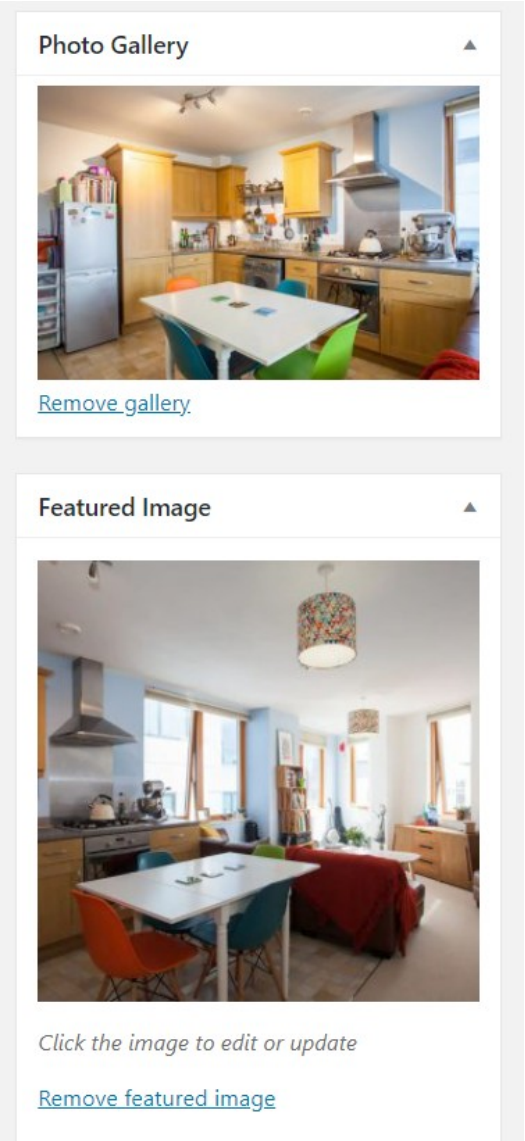
Στη συνέχεια μπορεί να επιλέξει πόλη αλλά να προσθέσει και μία καινούρια ενώ το ίδιο ισχύει και για τις παροχές του χώρου και το πόσους επισκέπτες μπορεί να φιλοξενήσει συνολικά.

Επιπλέον μπορεί να δηλώσει κάποια επιπλέον σερβις που προσφέρει ο χώρος πάλι με τη χρήση checkbox.



Εικόνα 87: Η φόρμα επιλογής επιπλέον υπηρεσιών

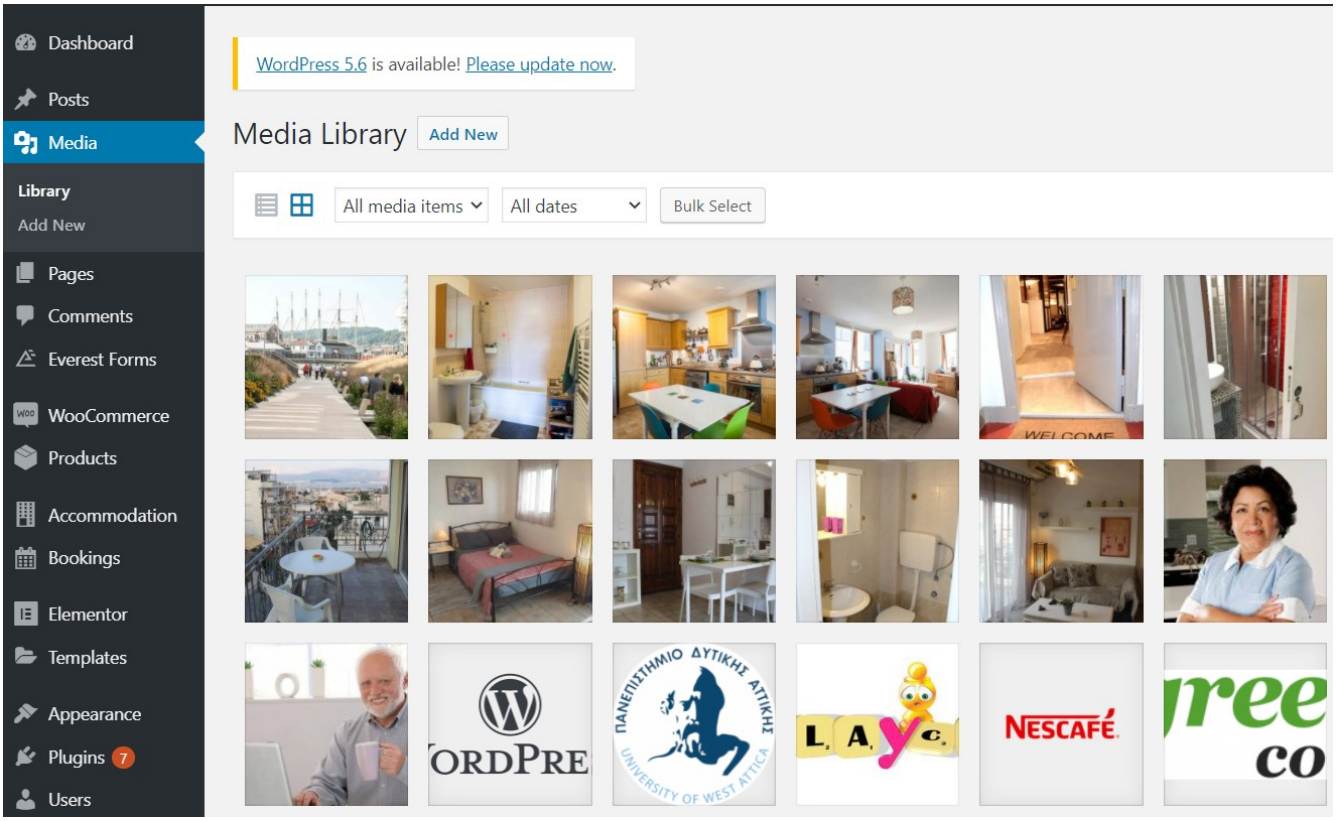
Οι admins μπορούν να επεξεργαστούν και τις φωτογραφίες που απεικονίζουν κάθε χώρο, ορίζοντας ποιά θα είναι η κεντρική φωτογραφία που θα δει ένας χρήστης ψάχνοντας τον χώρο αλλά και διαγράφοντας κάποιες από αυτές. Για να ανεβάσουν κάποια φωτογραφία στο σύστημα η διαδικασία που ακολουθείται δεν γίνεται σε αυτό το σημείο αλλά θα περιγραφεί στη συνέχεια.



Εικόνα 88: Οι φόρμες επιλογής εικόνων

Media Library

Όπως είδαμε οι admins είναι υπεύθυνοι για τις φωτογραφίες και γενικά το media υλικό που θα υπάρχει στο σύστημα άρα αυτοί θα ανεβάσουν τις φωτογραφίες στην εφαρμογή. Αυτό γίνεται μέσω της επιλογής Media, επιλέγοντας Add New ο admin μπορεί να προσθέσει μία νέα εικόνα ή και βίντεο, το οποίο στη συνέχεια μπορεί να χρησιμοποιήσει για κάποιο συγκεκριμένο χώρο ή ακόμα και για το design μίας σελίδας.

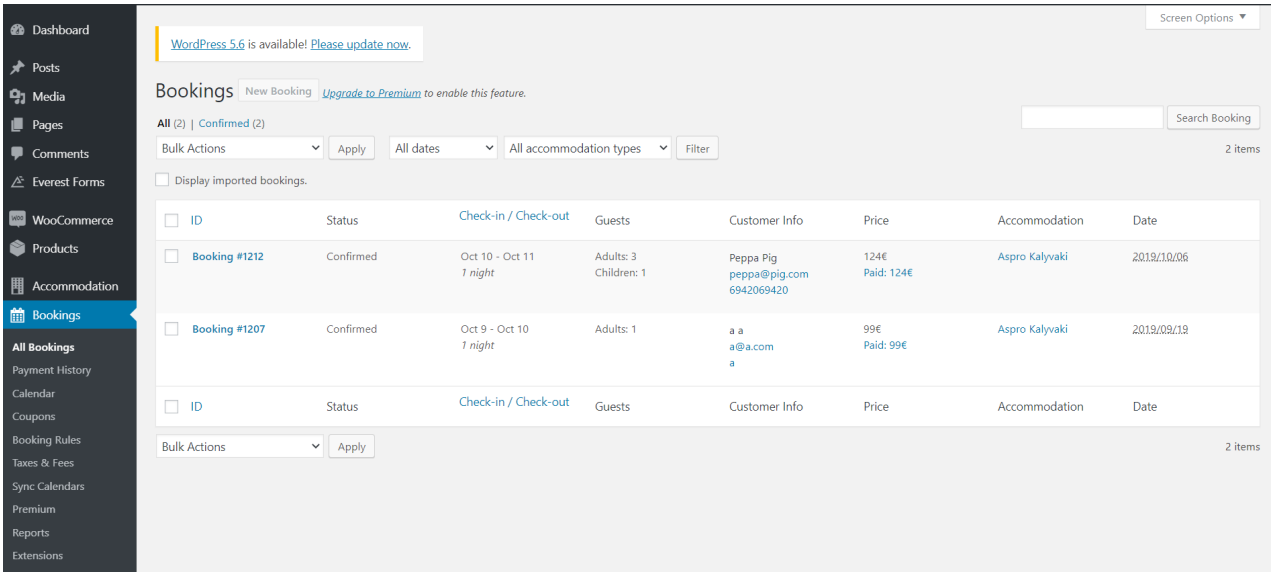


Εικόνα 89: Η σελίδα Media Library

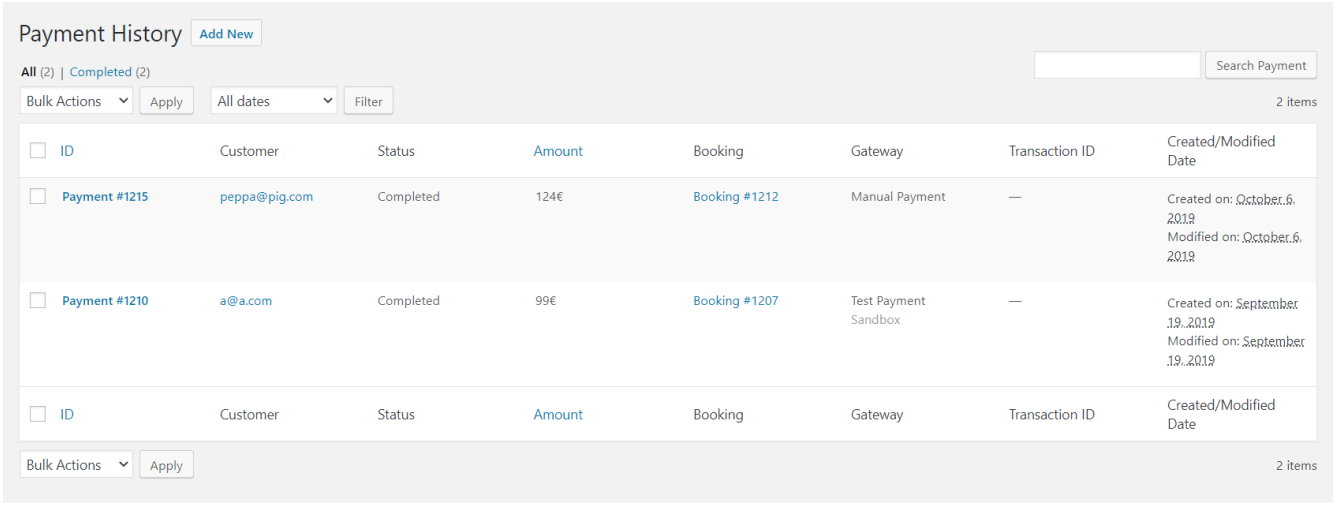
Bookings

Το booking είναι η βασική λειτουργία της εφαρμογής. Όπως είδαμε ο χρήστης για να πραγματοποιήσει την τελική κράτησή του και να μπορεί να εμφανιστεί αυτή στο σύστημα θα πρέπει πρώτα να δηλωθεί ότι έχει γίνει εξαργύρωση της κράτησης. Ωστόσο επειδή δεν κρίθηκε απαραίτητη η υλοποίηση αυτής της διαδικασία για να ολοκληρωθεί η κράτηση από έναν χρήστη, χρειάζεται να συνδεθεί ο admin και αφού ελέγξει τις νέες κρατήσεις που έχουν έρθει στο σύστημα να κατευθυνθεί στην επιλογή Payments όπου και πρέπει να δηλώσει ότι η πληρωμή έχει πραγματοποιηθεί. Όταν συμβεί αυτό η κράτηση μπορεί να επιβεβαιωθεί και ο χώρος θεωρείται κλεισμένος για τις ημερομηνίες που έχει επιλέξει ο χρήστης ενώ και άλλοι χρήστες μπορούν να δουν ότι το οίκημα δεν είναι διαθέσιμο για εκείνες τις ημερομηνίες. Τέλος, οι admins μπορούν να επιλέξουν τις ημερομηνίες για τις οποίες είναι ορατός ένας χώρος ενοικιάσεως.

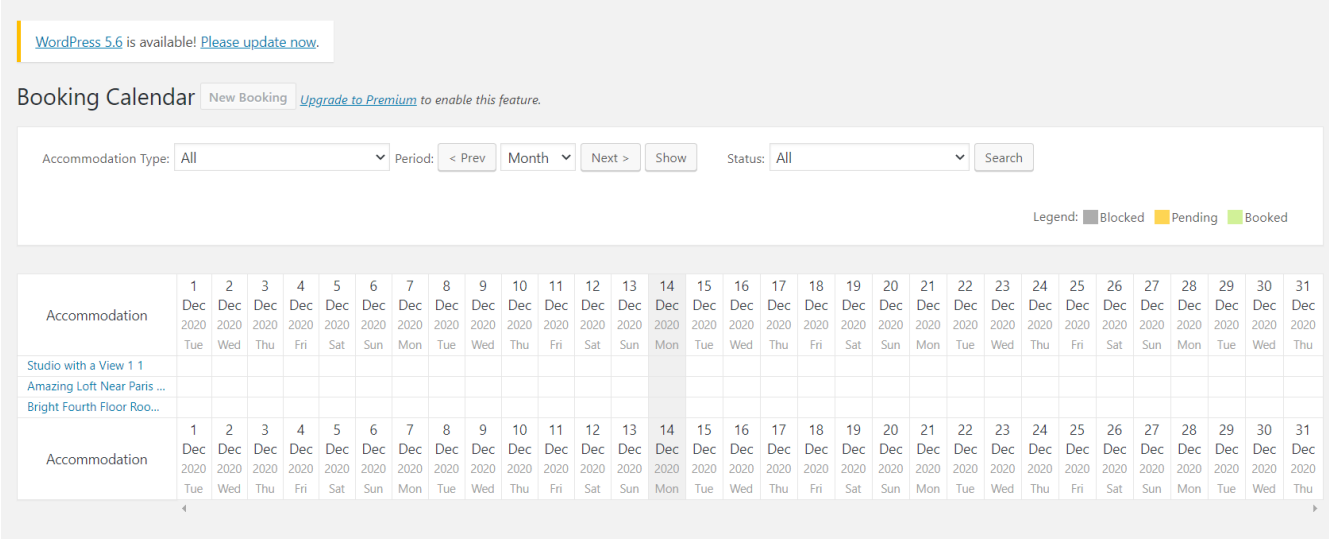
Οι εικόνες που παρουσιάζουν την παραπάνω διαδικασία είναι οι εξής και βλέπουμε όλες τις επιλογές που δίνονται στους admins.



Εικόνα 90: Η σελίδα προβολής των κρατήσεων

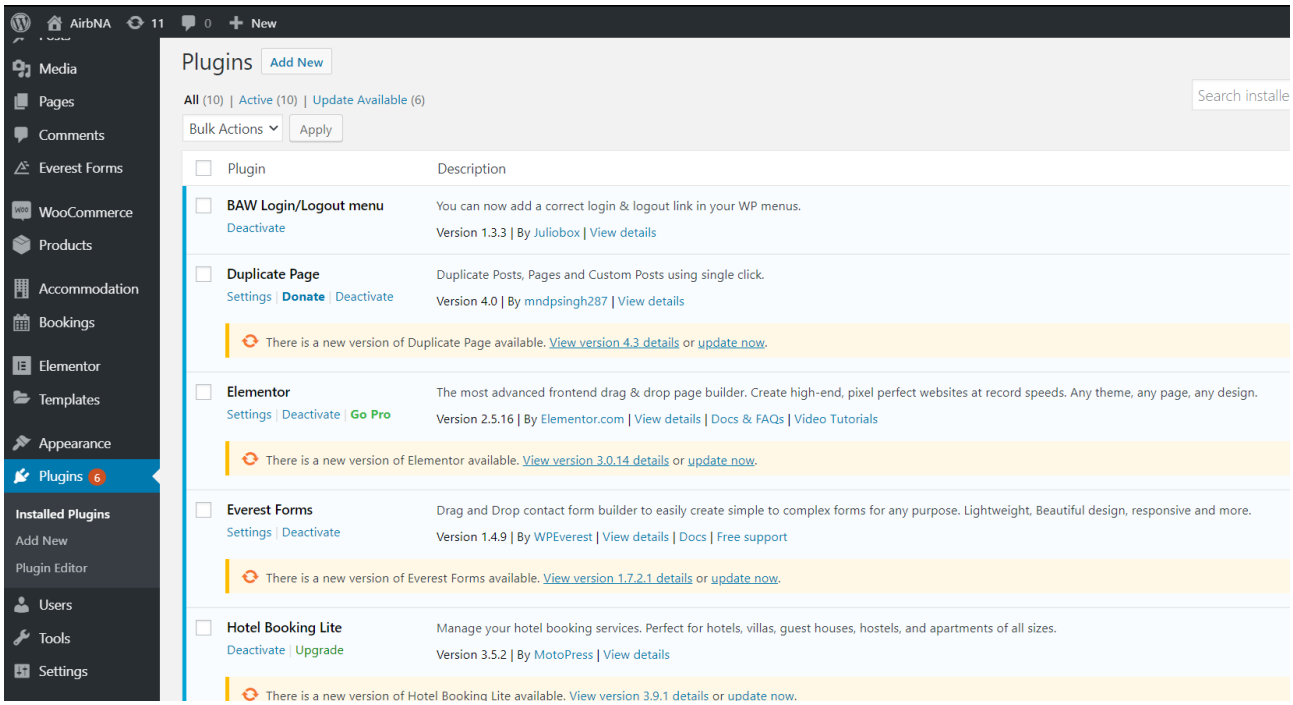


Εικόνα 91: Η σελίδα προβολής των πληρωμών των κρατήσεων



Εικόνα 92: Η σελίδα προβολής των ημερομηνιών των κρατήσεων

Plugins

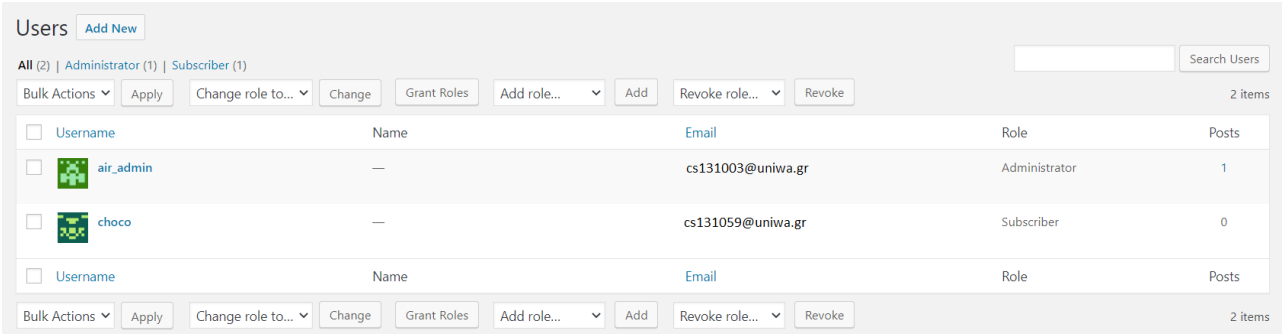


Εικόνα 93: Η σελίδα προβολής των plugins

Στην επιλογή Plugins οι admins μπορούν να δουν μία λίστα με όλα τα plugins που έχουν χρησιμοποιηθεί ήδη και να τα επεξεργαστούν. Μπορούν να αφαιρέσουν κάποιο από τα ήδη υπάρχοντα διαγράφοντας το είτε να προσθέσουν κάποιο καινούριο επιλέγοντας Add New. Μπορούν επίσης να ενημερώσουν τα plugins που έχουν ήδη χρησιμοποιηθεί κάνοντας update.

Users

Εδώ οι admins μπορούν να επεξεργαστούν τους ρόλους των χρηστών της εφαρμογής. Μπορούν να δώσουν δικαιώματα δηλαδή σε ένα χρήστη με βάση το ρόλο που θα του αναθέσουν και να ορίσουν ακόμα και νέους admins. Επιπλέον μπορούν να τροποποιήσουν τους ήδη υπάρχοντες ρόλους ή να διαγράψουν χρήστες από την εφαρμογή.



Εικόνα 94: Η σελίδα προβολής των χρηστών

Η βάση δεδομένων της εφαρμογής δημιουργείται αυτόματα από το ίδιο το Wordpress. Αυτό αποτελεί τόσο πλεονέκτημα όσο και μειονέκτημα για τον προγραμματιστή. Παρότι εξοικονομείται χρόνος καθώς δεν χρειάζεται να μπει στη διαδικασία δημιουργίας της βάσης υπάρχουν προβλήματα που έχουν να κάνουν με τη συντήρηση της καθώς είναι αρκετά πιο πολύπλοκη για να την κατανοήσει.

6.3 Node.js

Η τρίτη κατά σειρά τεχνική υλοποίησης που χρησιμοποιήθηκε για την δημιουργία εφαρμογής τύπου Airbnb ήταν η Node.js. Όπως αναφέρθηκε είναι βασισμένη σε τεχνοτροπία MVC και η δομή της θα παρουσιαστεί στη συνέχεια. Η βάση που χρησιμοποιήθηκε είναι τύπου MongoDB.

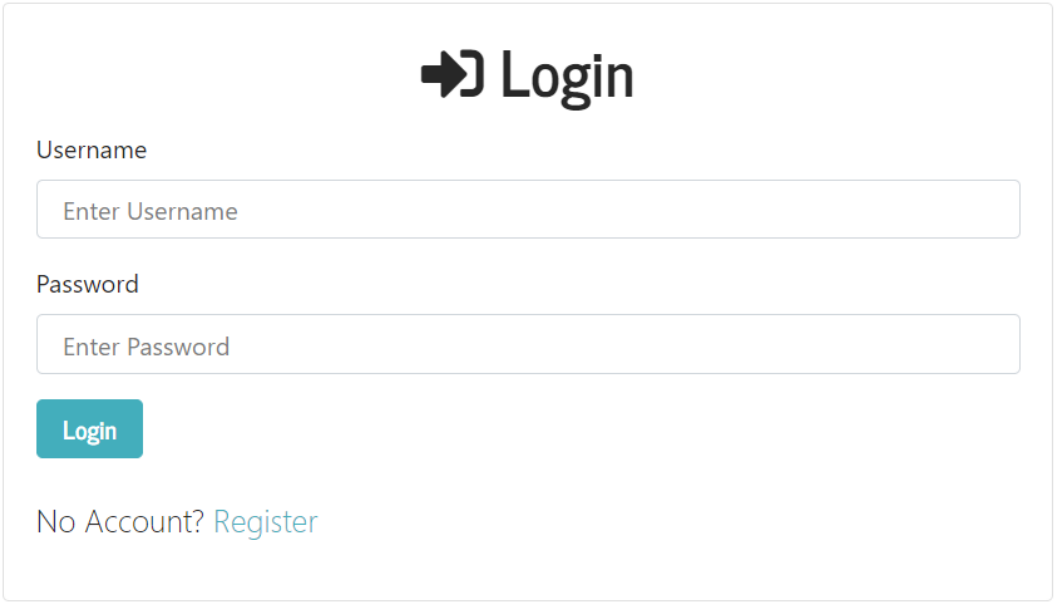
Front End

Εικόνα 95: Η αρχική σελίδα της εφαρμογής Node.js

Οι χρήστες σαν πρώτη σελίδα της εφαρμογής θα δουν μία σελίδα στην οποία παρουσιάζεται το logo ενώ στην μπάρα του μενού στο πάνω μέρος της σελίδας εμφανίζονται και οι επιλογές για login που αφορά όσους έχουν ήδη δώσει τα στοιχεία τους και μπορούν να συνδεθούν συμπληρώνοντας τα στοιχεία που θα τους ζητηθούν μαζί με τον κωδικό τους ή η επιλογή register που αφορά νέους χρήστες της εφαρμογής που θέλουν να καταχωρήσουν για πρώτη φορά τα στοιχεία τους ούτως ώστε να κάνουν χρήστη αυτής.

Επιλέγοντας το “Login” ο χρήστης μεταφέρεται στη σελίδα login στην οποία θα δει ένα παράθυρο στο οποίο θα του ζητηθεί να καταχωρήσει τα προσωπικά του στοιχεία, username και password. Αν κάποιος χρήστης λανθασμένα επιλέξει την επιλογή login μπορεί μέσω του link register να ανακατευθυνθεί στη σωστή σελίδα.

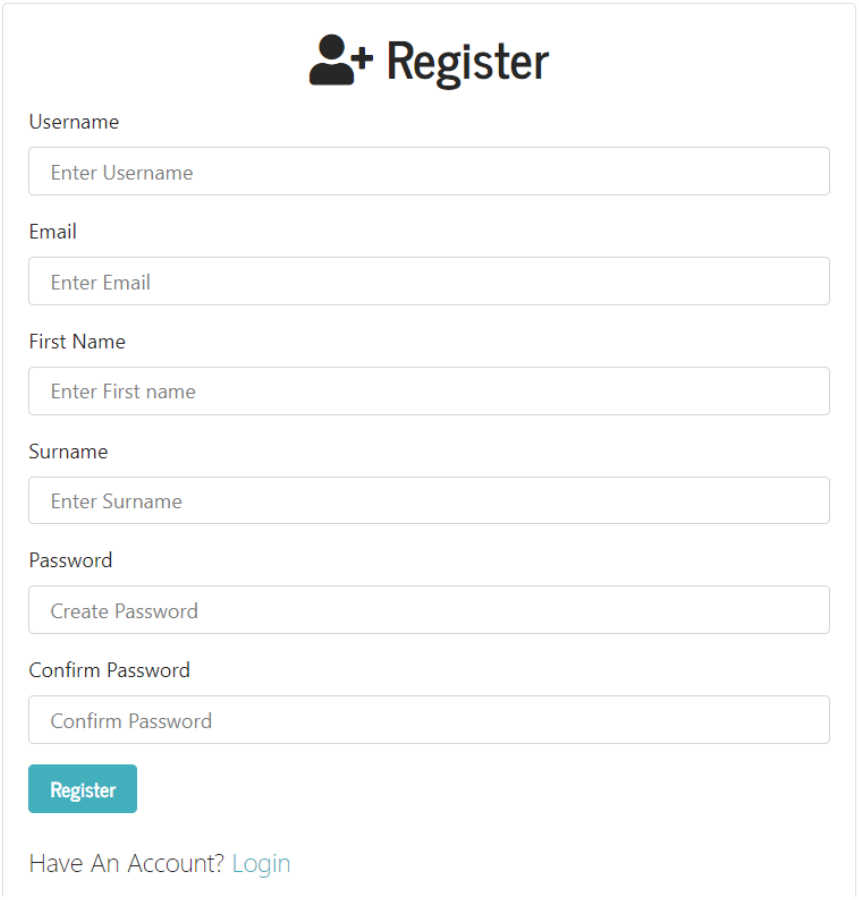
Login – Register



Εικόνα 96: Η φόρμα Login

Τοποθετώντας τα σωστά στοιχεία σε αυτή τη φόρμα ο χρήστης μπορεί να συνδεθεί με την εφαρμογή.

Εφόσον όπως αναφέρθηκε δεν έχει πραγματοποιήσει εγγραφή ήδη στην εφαρμογή ο χρήστης τότε θα επιλέξει την επιλογή register. Μεταφέρεται έτσι στη σελίδα register στην οποία θα πρέπει να συμπληρώσει τα ανάλογα στοιχεία που θα του ζητηθούν στα αντίστοιχα πεδία. Αντίστοιχα με τη σελίδα login αν έχει βρεθεί εδώ κατά λάθος ένας χρήστης ενώ έχει πραγματοποιήσει ήδη εγγραφή μπορεί να μεταφερθεί άμεσα στη σελίδα login.



Εικόνα 97: Η φόρμα Register

Admin/User Dashboard

Πραγματοποιώντας είσοδο στην εφαρμογή ένας χρήστης διακρίνουμε ότι στην μπάρα επιλογών εμφανίζονται και οι επιλογές Profile και Create Place.

Όταν ο χρήστης επιλέξει την επιλογή Profile θα ανακατευθυνθεί στο dashboard όπου μπορεί να δει όλα τα στοιχεία του αλλά και τις επιλογές που έχει. Αν είναι admin τότε οι επιλογές του διαφέρουν από αυτές που έχει ένας απλός user. Οι διαφορές αυτές φαίνονται στο admin panel όπου έχει επιλογές για τη δημιουργία πόλης, παροχής και χώρου διαμονής. Όλοι οι χρήστες μπορούν αν δουν τους δικούς τους χώρους, αυτούς που αυτοί παρέχουν καθώς και τις κρατήσεις που έχουν πραγματοποιήσει.

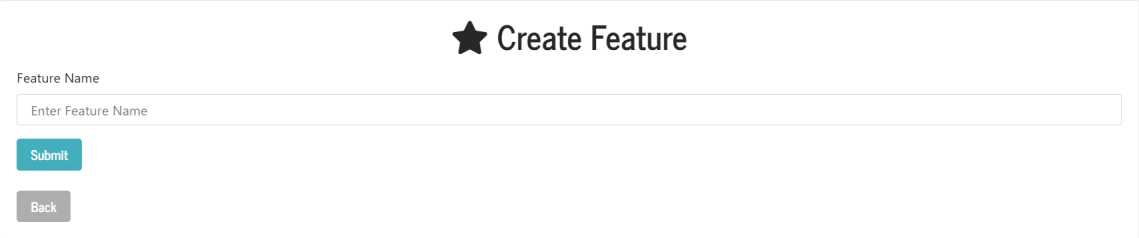
Εικόνα 98: Η σελίδα dashboard για έναν admin της εφαρμογής

Εικόνα 99: Η σελίδα dashboard για έναν user της εφαρμογής

Create City – Feature

Όπως αναφέρθηκε ο Admin έχει την επιλογή να δημιουργήσει μία πόλη ή μία παροχή. Επιλέγοντας από το dashboard την αντίστοιχη επιλογή ανακατευθύνεται και στην ανάλογη σελίδα όπου θα παρουσιαστεί μπροστά του μια φόρμα με στοιχεία που πρέπει να συμπληρώσει ούτως ώστε να προστεθεί η πληροφορία που θέλει στην εφαρμογή. Αυτό πραγματοποιείται αποθηκεύοντας την πληροφορία στη βάση.

Εικόνα 100: Η σελίδα δημιουργίας ενός city

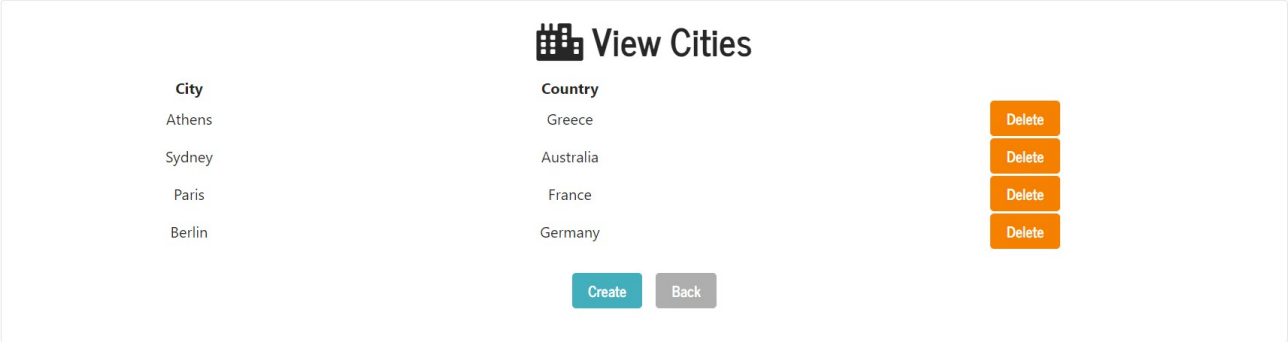


Εικόνα 101: Η σελίδα δημιουργίας ενός feature

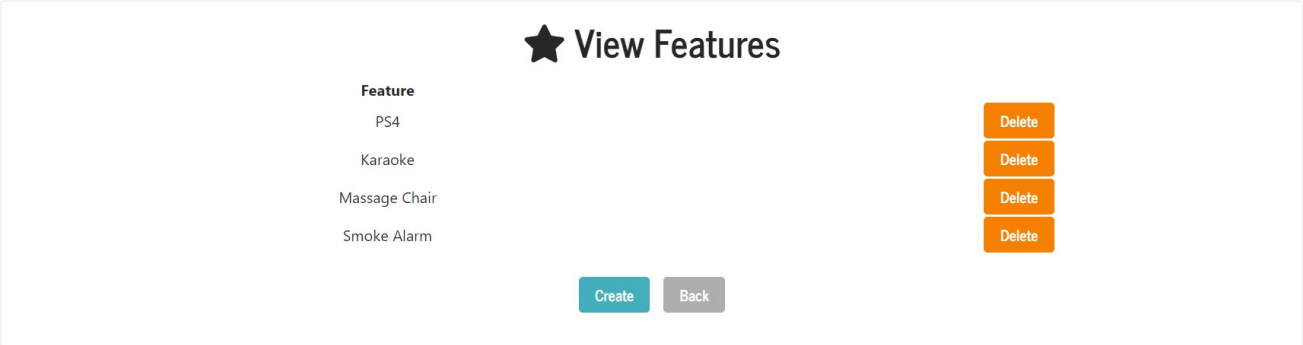
Delete City – Feature

Αντίστοιχα με τη δημιουργία ο Admin μπορεί να διαγράψει μία πληροφορία από την εφαρμογή μέσω των επιλογών delete. Έτσι ο admin ανάλογα με την επιλογή του μπορεί να διαγράψει μία πόλη ή ένα feature.

Στην κάθε σελίδα εμφανίζεται μία λίστα με τις πόλεις ή τα features που υπάρχουν ήδη και δίπλα στο όνομα του καθενός ο admin βλέπει την επιλογή delete που εκτελώντας την, την διαγράφει από την εφαρμογή και ως αποτέλεσμα από τη βάση τη συγκεκριμένη πληροφορία.



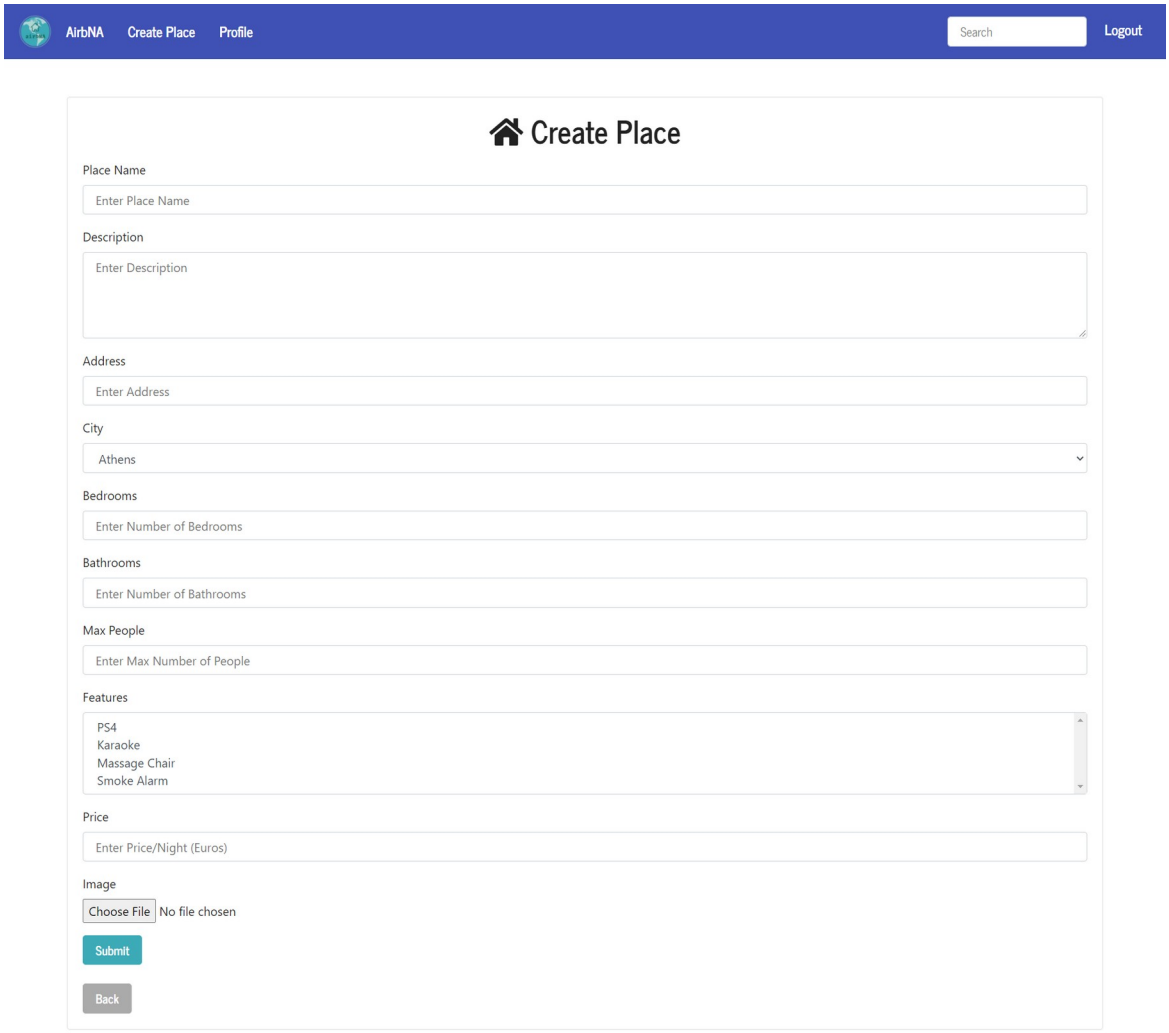
Εικόνα 102: Η σελίδα προβολής/διαγραφής των cities



Εικόνα 103: Η σελίδα προβολής/διαγραφής των features

Create City

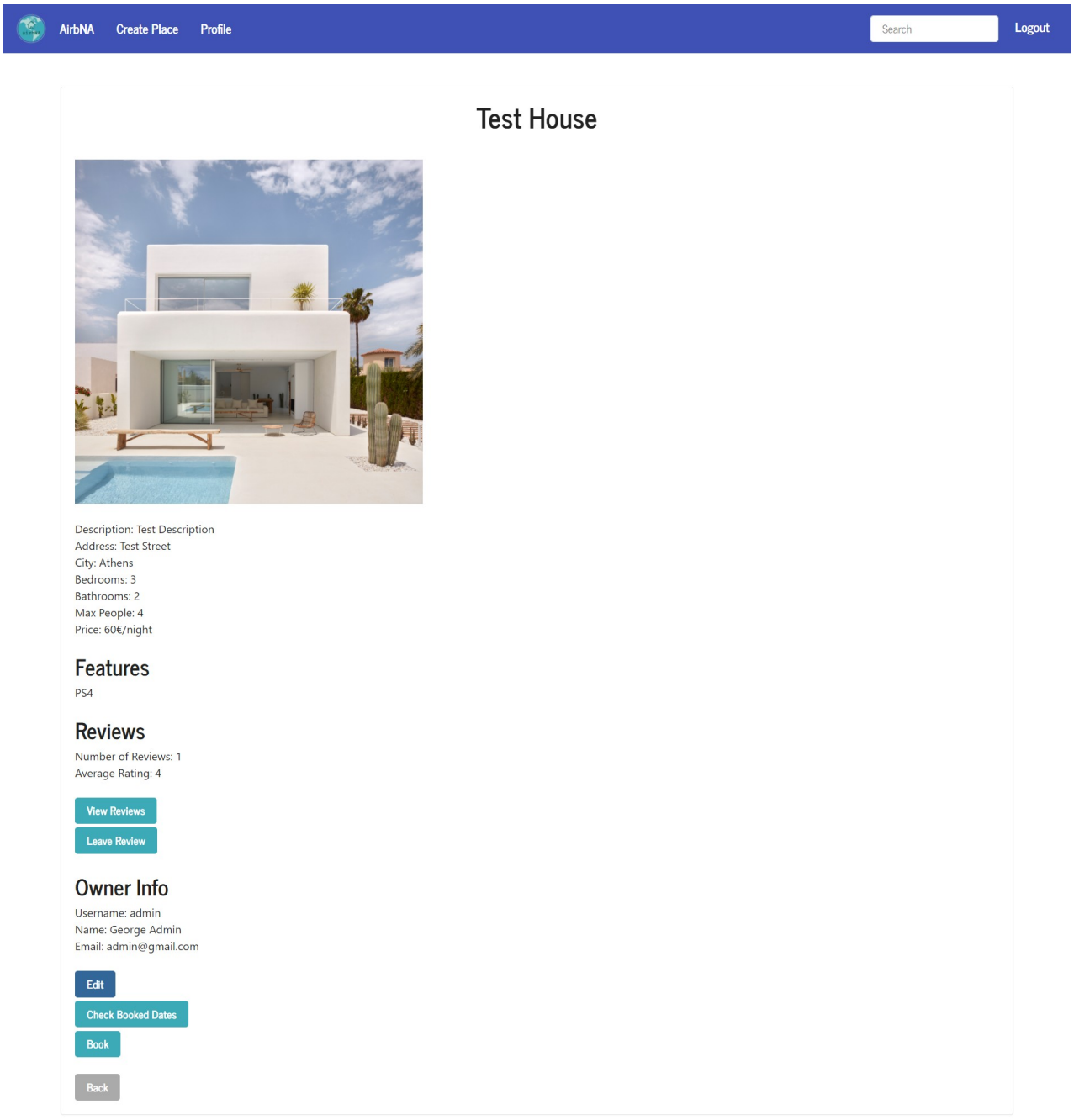
Τη δυνατότητα δημιουργία χώρου (place) έχουν όλοι οι χρήστες της εφαρμογής.



Εικόνα 104: Η σελίδα δημιουργίας ενός place

Για να δημιουργηθεί ένας χώρος ο χρήστης πρέπει να συμπληρώσει όλα τα πεδία με τις απαραίτητες πληροφορίες. Το πεδίο city δίνει την επιλογή στο χρήστη να διαλέξει από τις ήδη υπάρχουσες πόλεις καθώς εμφανίζονται σε μία λίστα ενώ στο πεδίο features μπορεί να διαλέξει πολλαπλές παροχές.

View Place



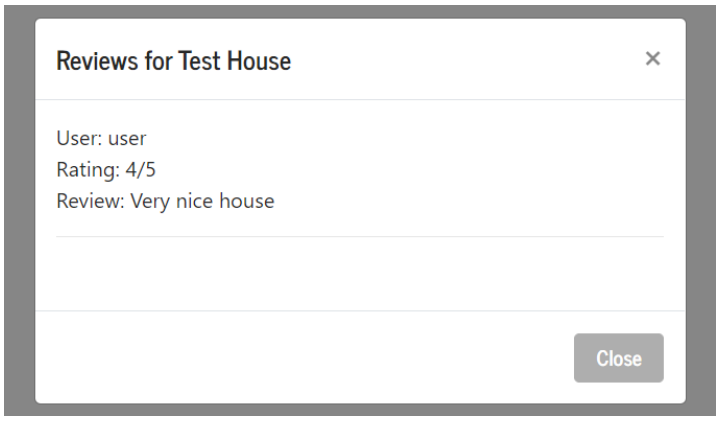
Εικόνα 105: Η σελίδα προβολής ενός συγκεκριμένου place

Στη σελίδα αυτή οι χρήστες μπορούν αν δουν αναλυτικές πληροφορίες για έναν χώρο. Αρχικά βλέπουν την βασική εικόνα που αντιπροσωπεύει το χώρο ενώ στη συνέχεια ακολουθούν στοιχεία που έχουν αν κάνουν με το πόσα άτομα μπορεί να φιλοξενήσει, σε ποια πόλη βρίσκεται, η τιμή του και τα δωμάτια που περιέχει. Ο χρήστης μπορεί επίσης να δει αναλυτικά τις παροχές που περιλαμβάνει ο χώρος.

Επιπλέον μπορεί να βρει τις απόψεις άλλων χρηστών και τη βαθμολογία που αυτοί έδωσαν για την εμπειρία τους εκεί (Reviews). Δίνεται η επιλογή στον χρήστη να προσθέσει και τη δική του κριτική (Leave Review). Τέλος εμφανίζονται πληροφορίες για τον ιδιοκτήτη και οι δυνατότητες να ελέγξει κάποιος τις ημερομηνίες κράτησης (Check Booked Dates) και εφόσον το επιθυμεί να πραγματοποιήσει μία (Book). Αν ο χρήστης είναι ο admin ή ο ιδιοκτήτης του χώρου τότε έχει την επιλογή να πραγματοποιήσει αλλαγές στα στοιχεία του (Edit).

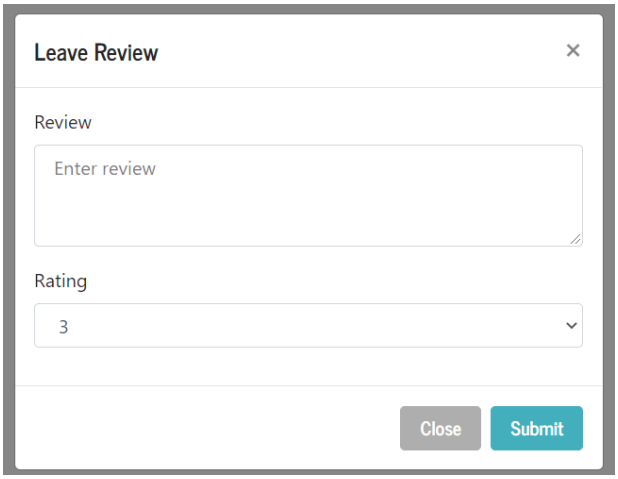
Reviews

Αν ο χρήστης επιλέξει να δει τα σχόλια που έχουν αφήσει άλλοι χρήστες για τον χώρο που τον ενδιαφέρει τότε κατευθύνεται σε άλλη σελίδα όπου και βλέπει το username του χρήστη που άφησε το σχόλιο, τη βαθμολογία του και το σχόλιο που άφησε.



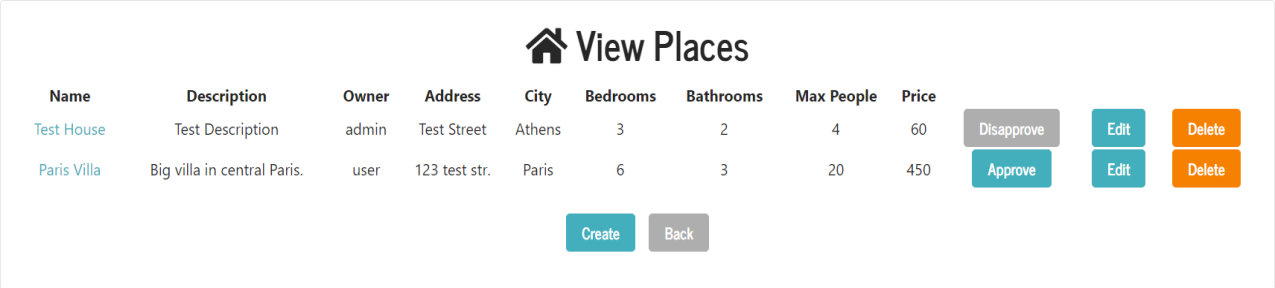
Εικόνα 106: Το παράθυρο modal προβολής των review

Ενώ αν επιλέξει να αφήσει ένα σχόλιο τότε θα εμφανιστεί παράθυρο στο οποίο υπάρχει πεδίο για να συμπληρώσει την άποψη του και στη συνέχεια, μπορεί να επιλέξει μία βαθμολογία που θέλει να αφήσει μεταξύ του 1 και του 5.



Εικόνα 107: Το παράθυρο modal δημιουργίας ενός review

View Places

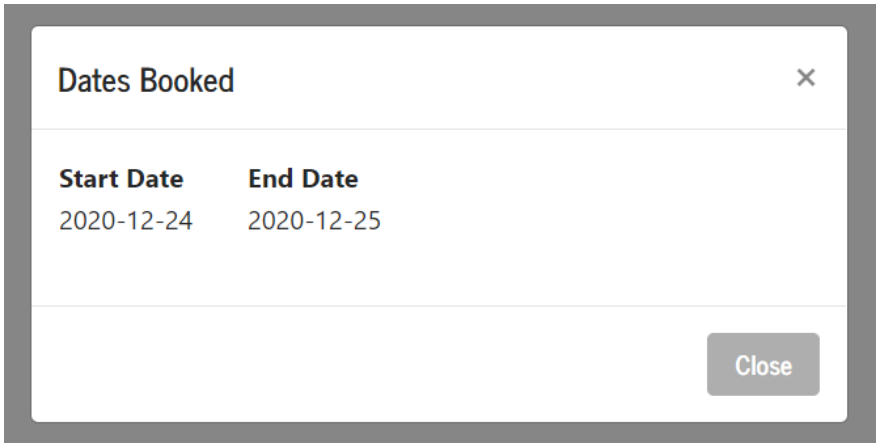


Εικόνα 108: Η σελίδα προβολής των place με επιπλέον επιλογές

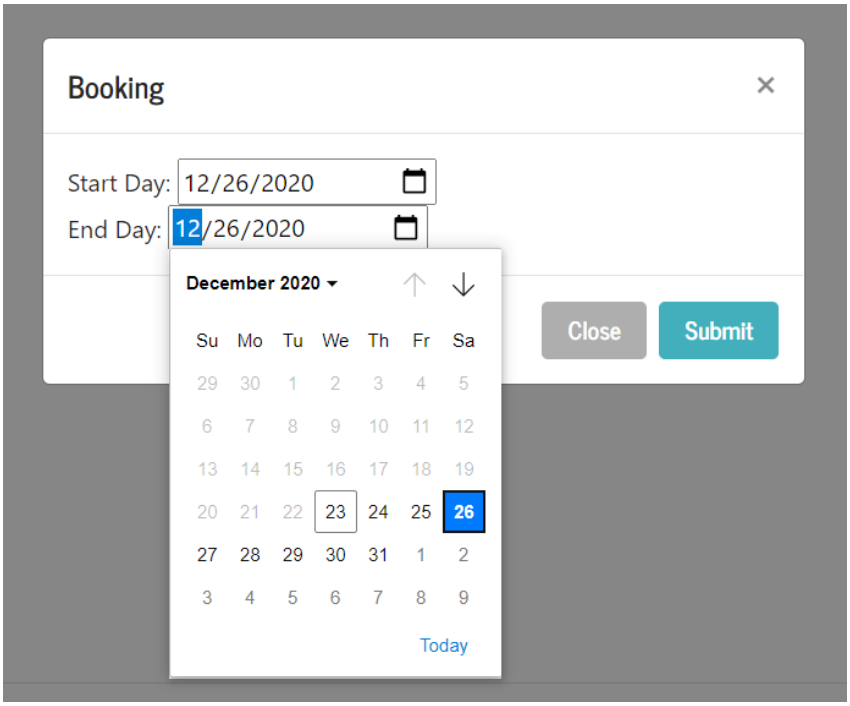
Πρόσβαση στα στοιχεία αυτά έχουν μόνο οι Admins όπου μπορούν να δουν τα places που περιμένουν στη σειρά για να εγκριθούν ούτως ώστε να γίνουν διαθέσιμα. Εδώ μπορεί επίσης να επιλέξει απευθείας να πραγματοποιήσει τις τροποποιήσεις που θέλει ή να διαγράψει το χώρο.

Booking

Για να πραγματοποιήσει μία κράτηση ένας χρήστης μπορεί αρχικά να ελέγξει ποιες ημερομηνίες είναι ήδη κλεισμένο όπως είδαμε και στο View Place. Αφού ελέγξει τις ημερομηνίες μπορεί να επιλέξει πότε θέλει ο ίδιος να πραγματοποιήσει κράτηση και να υποβάλει την επιλογή του.

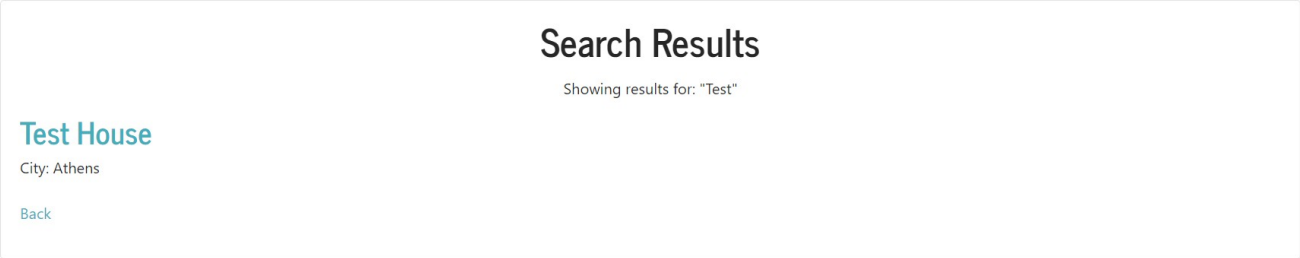


Εικόνα 109: Το παράθυρο modal για τη προβολή των μη διαθέσιμων ημερομηνιών



Εικόνα 110: Το παράθυρο modal για την υποβολή ενός booking

Search Results

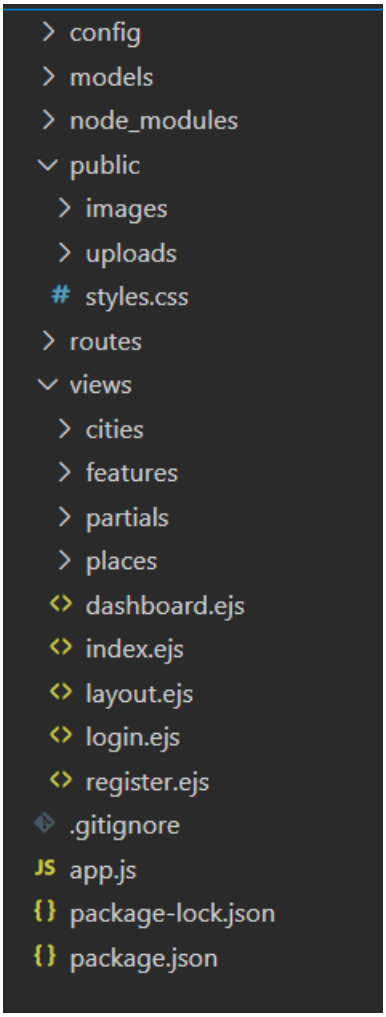


Εικόνα 111: Η σελίδα Search Results

Οι χρήστες μπορούν να πραγματοποιήσουν αναζήτηση από την επιλογή που δίνεται στην μπάρα. Μπορούν να ψάξουν με βάση την πόλη ή τον τίτλο του χώρου και σαν αποτέλεσμα να πάρουν τους χώρους που ανταποκρίνονται στα στοιχεία αυτά.

Back End

Στο προηγούμενο κεφάλαιο παρουσιάστηκε η εφαρμογή με βάση τη NodeJs. ο κώδικας που δημιουργεί τις σελίδες αλλά και τη λειτουργία που περιγράφτηκε θα παρουσιαστεί σε αυτό το κεφάλαιο.



Εικόνα 112: Η δομή του κώδικα της εφαρμογής

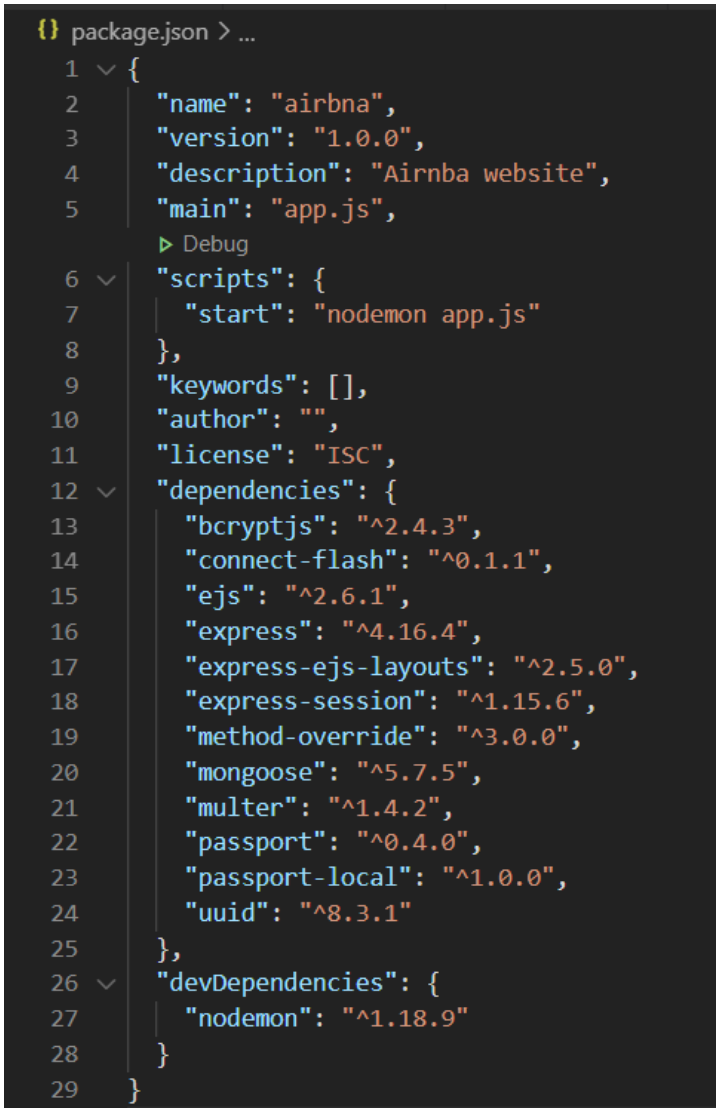
Όπως έχει επίσης αναφερθεί ο κώδικας είναι γραμμένος σύμφωνα με το μοντέλο MVC (ο τρόπος λειτουργίας του οποίου έχει αναπτυχθεί στο κεφάλαιο τρία, παράγραφος πέντε) ενώ η βάση που χρησιμοποιήθηκε είναι η MongoDB.

Στην συγκεκριμένη περίπτωση αντί για controllers υπάρχει ο φάκελος routes μέσα στον οποίον έχουμε τα αρχεία που σε άλλη περίπτωση θα ήταν στον φάκελο controllers εδώ όμως χρησιμοποιήθηκε η ονομασία routes.

Package.json

Το αρχείο package.json περιέχει τις βασικές πληροφορίες της εφαρμογής καθώς και όλα τα πακέτα που χρησιμοποιούμε. Ουσιαστικά είναι τα έτοιμα εργαλεία που χρειάζονται για να δουλέψει ο κώδικας όπως στην PHP είχαμε τις βιβλιοθήκες ή τα plugins του wordpress, αντίστοιχα εδώ έχουμε τα πακέτα.

Κάποια από τα βασικά πακέτα που χρησιμοποιήθηκαν είναι το **mongoose** που επιτρέπει τη σύνδεση με τη βάση δεδομένων, το **express**, το οποίο επιτρέπει την εύκολη δημιουργία web application και API, και το **passport** που είναι υπεύθυνο για τον έλεγχο της διεκπεραίωσης των λειτουργιών register, login και user authentication.



Εικόνα 113: Το αρχείο package.json

App.js

Είναι το βασικό αρχείο της εφαρμογής. Εδώ φορτώνονται και στήνονται όλα τα πακέτα που χρησιμοποιούνται, τα routes καθώς γίνεται και η δήλωση της πόρτας στην οποία απαντάει η εφαρμογή και η σύνδεση με τη βάση MongoDB.

app.js > ...

```
1   // Library/Package requirements
2   const express = require('express');
3   const expressLayouts = require('express-ejs-layouts');
4   const mongoose = require('mongoose');
5   const passport = require('passport');
6   const flash = require('connect-flash');
7   const session = require('express-session');
8   const methodOverride = require('method-override');
9
10  const app = express();
11
12  // Passport package config
13  require('./config/passport')(passport);
14
15  // Connect to MongoDB
16  mongoose.connect('mongodb://localhost/airbna', { useNewUrlParser: true, useUnifiedTopology: true })
17
18  // EJS setup
19  app.use(expressLayouts);
20  app.set('view engine', 'ejs');
21
22  // Express body parser
23  app.use(express.urlencoded({ extended: true }));
24
25  // Setup method override for POST -> DELETE
26  app.use(methodOverride('_method'))
27
28  // Express session
29  app.use(
30    session({
31      secret: 'secret',
32      resave: true,
33      saveUninitialized: true
34    })
35  );
36
```

Εικόνα 114: Ο κώδικας του αρχείου app.js

```
// Passport middleware
app.use(passport.initialize());
app.use(passport.session());

// Connect flash
app.use(flash());

// Global variables
app.use(function(req, res, next) {
  res.locals.success_msg = req.flash('success_msg');
  res.locals.error_msg = req.flash('error_msg');
  res.locals.error = req.flash('error');
  next();
});

// Public folder
app.use(express.static(__dirname+'/public'));

// Routes
app.use('/', require('./routes/index.js'));
app.use('/users', require('./routes/users.js'));
app.use('/cities', require('./routes/cities.js'));
app.use('/features', require('./routes/features.js'));
app.use('/places', require('./routes/places.js'));
app.use('/book', require('./routes/bookings.js'));
app.use('/review', require('./routes/reviews.js'));

// Last Route to catch unused routes
app.all('*', function(req, res) {
  res.redirect("/");
});

// Setup server port
const PORT = process.env.PORT || 5000;
app.listen(PORT, console.log('Server started on port', PORT));
```

Εικόνα 115: Η συνέχεια του κώδικα του αρχείου app.js

Config

Ο κώδικας που υπάρχει στα αρχεία του config έχει να κάνει με το configuration ενός χρήστη στην εφαρμογή. Στο αρχείο **auth.js** υπάρχουν συναρτήσεις που ελέγχουν αν ο χρήστης είναι ήδη συνδεδεμένος, τον ρόλο του χρήστη και εάν είναι admin και εάν ο χρήστης δεν είναι συνδεδεμένος.

config > JS auth.js > ...

```
1   module.exports = {
2
3     // Function to check if user is logged in
4     ensureAuthenticated: function(req, res, next) {
5       if (req.isAuthenticated()) {
6         return next();
7       }
8       req.flash('error_msg', 'Please log in to view that resource');
9       res.redirect('/users/login');
10    },
11
12    // Function to check if user is logged in and has 'admin' role
13    ensureAdmin: function(req, res, next) {
14      if (req.isAuthenticated() && req.user.role === 'admin') {
15        return next();
16      }
17      req.flash('error_msg', 'Not authorized to view that resource');
18      res.redirect('/');
19    },
20
21    // Function to check if user is not logged in
22    forwardAuthenticated: function(req, res, next) {
23      if (!req.isAuthenticated()) {
24        return next();
25      }
26      res.redirect('/dashboard');
27    }
28
29  };
```

Εικόνα 116: Ο κώδικας του αρχείου config.js

Στο αρχείο **passport.js** ο κώδικας που υπάρχει ορίζει τον τρόπο λειτουργίας του login. Θέτει δηλαδή σαν κανόνα ότι για να πραγματοποιηθεί το login παρακάτω ενός χρήστη ο χρήστης αυτός θα πρέπει να έχει δώσει σαν όνομα χρήστη και κωδικό ένα όνομα χρήστη και έναν κωδικό που υπάρχουν αποθηκευμένα στη βάση.

```
config > JS passport.js > ...
1   const LocalStrategy = require('passport-local').Strategy;
2   const bcrypt = require('bcryptjs');
3
4   // Load User model
5   const User = require('../models/User');
6
7   module.exports = function(passport) {
8     passport.use(
9       new LocalStrategy({ usernameField: 'username' }, (username, password, done) => {
10        // Match user
11        User.findOne({
12          username: username
13        }).then(user => {
14          if (!user) {
15            return done(null, false, { message: 'Incorrect Username/Password' });
16          }
17
18          // Match password
19          bcrypt.compare(password, user.password, (err, isMatch) => {
20            if (err) throw err;
21            if (isMatch) {
22              return done(null, user);
23            } else {
24              return done(null, false, { message: 'Incorrect Username/Password' });
25            }
26          });
27        });
28      })
29    );
30
```

Εικόνα 117: Ο κώδικας στησήματος του plugin Passport

Αν και εφόσον πραγματοποιηθεί σωστά το login τότε χρησιμοποιούνται οι συναρτήσεις serializeUser και deserializeUser για την δημιουργία session στα οποία υπάρχει η πληροφορία για τον συνδεδεμένο user καθώς θα χρειαστεί στη συνέχεια να γνωρίζει το σύστημα ποιος user είναι συνδεδεμένος τη δεδομένη χρονική στιγμή.

```
31    passport.serializeUser(function(user, done) {
32      done(null, user.id);
33    });
34
35    passport.deserializeUser(function(id, done) {
36      User.findById(id, function(err, user) {
37        done(err, user);
38      });
39    });
40  };
```

Εικόνα 118: Ο κώδικας των συναρτήσεων του Passport

Models

Στο φάκελο models υπάρχει ο κώδικας που είναι υπεύθυνος για τη σύνδεση με τη βάση και τη δημιουργία των απαραίτητων μοντέλων με βάση τα οποία θα γίνει η επικοινωνία με τα documents της βάσης. Χρειάζονται για να κληθούν πληροφορίες από τη βάση ή για να αποθηκευτούν στη βάση νέα στοιχεία που θα δώσουν οι χρήστες στην εφαρμογή.

Για παράδειγμα το model που αφορά τα places έχει την εξής δομή:

models > JS Place.js > ...

```
1   const mongoose = require('mongoose');
2
3   // Model for Place schema in MongoDB
4   const PlaceSchema = new mongoose.Schema({
5     name: { type: String, required: true },
6     description: { type: String },
7     owner: { type: String },
8     address: { type: String },
9     city: { type: String },
10    bedrooms: { type: Number, min: 0 },
11    bathrooms: { type: Number, min: 0 },
12    max_people: { type: Number, min: 0 },
13    price: { type: Number, min: 0 },
14    approved: { type: Boolean, default: false },
15    features: { type: [String] },
16    image: { type: String }
17  });
18
19  const Place = mongoose.model('Place', PlaceSchema);
20
21  module.exports = Place;
```

Εικόνα 119: Παράδειγμα κώδικα ενός μοντέλου της εφαρμογής

Παρομοίως δημιουργούνται και τα models για τις υπόλοιπες λειτουργίες που είναι απαραίτητη και η επικοινωνία με τη βάση. Αυτά είναι το model για booking, city, feature, review και user.

Routes

Στα routes περιλαμβάνονται τα αρχεία αυτά που περιέχουν τον κώδικα ο οποίο πραγματοποιεί τη σύνδεση του interface της εφαρμογής με τη βάση ενώ προσδίδει λειτουργία σε αυτόν. Αν για παράδειγμα υπάρχει ένα κουμπί που ορίστηκε στο interface και με βάση το οποίο θα διαγραφεί μία πόλη από τη βάση για να πραγματοποιηθεί η διαγραφή αυτή καθώς και για να οριστεί ότι είναι το κουμπί που διαγράφει την πόλη χρησιμοποιείται ο κώδικας που βρίσκεται στο αντίστοιχο αρχείο route.

Places

routes > JS places.js > ...

```
1   const express = require('express');
2   const router = express.Router();
3   var multer = require('multer');
4
5   // Load models and functions
6   const Place = require('../models/Place');
7   const City = require('../models/City');
8   const Feature = require('../models/Feature');
9   const User = require('../models/User');
10  const Booking = require('../models/Booking');
11  const Review = require('../models/Review');
12  const {
13      forwardAuthenticated,
14      ensureAdmin,
15      ensureAuthenticated
16  } = require('../config/auth');
17  const {
18      Mongoose,
19      Schema
20  } = require('mongoose');
21
```

Εικόνα 120: Ο κώδικας φόρτισης των model και συναρτήσεων

Αρχικά περιλαμβάνονται τα απαραίτητα model ούτως ώστε να πραγματοποιηθούν οι λειτουργίες για τα places.

Για να πραγματοποιηθεί η λειτουργία **upload image** γίνεται χρήση του πακέτου multer. Δηλώνεται σε ποιον φάκελο θέλουμε να αποθηκευτούν οι εικόνες και με ποιον τρόπο και στη συνέχεια γίνεται κλήση του πακέτου.

```
22  // Image Uploading
23  var storage = multer.diskStorage({
24      destination: function (req, file, cb) {
25          cb(null, 'public/uploads/')
26      },
27      filename: function (req, file, cb) {
28          cb(null, file.originalname);
29      }
30  });
31  var upload = multer({
32      storage: storage
33  });
```

Εικόνα 121: Ο κώδικας στησήματος του plugin Multer

Για την λειτουργία **View Places** πραγματοποιείται το αντίστοιχο query στη βάση που ψάχνει με βάση το place που δηλώσαμε ότι θέλουμε και ελέγχει αν υπάρχει αντίστοιχο δεδομένο αποθηκευμένο σε αυτή. Εάν υπάρχει το βρίσκει και το επιστρέφει για να εμφανιστεί στο χρήστη.

```
84  // View Places
85  router.get('/list', ensureAdmin, (req, res, next) => {
86      Place.find()
87          .exec()
88          .then(docs => {
89              res.render('places/list', {
90                  placeData: docs,
91                  user: req.user
92              });
93          })
94          .catch(err => {
95              console.log(err);
96              res.status(500).json({
97                  error: err
98              });
99          });
100 });
```

Εικόνα 122: Ο κώδικας υλοποίησης της προβολής των place

Για τις λειτουργίες **approve place** και **disapprove place** γίνεται χρήση δύο router συναρτήσεων που δέχονται ως είσοδο το ζητούμενο του χρήστη ενώ επιστρέφουν την απάντηση που θα πάρει αυτός. Στη σύνδεση με το αντίστοιχο model της βάσης βλέπουμε ότι γίνεται χρήση του query updateOne το οποίο βρίσκει ένα place με το αντίστοιχο id που έδωσε ο χρήστης στη βάση και ανάλογα τη λειτουργία ενημερώνει την εγγραφή της βάσης ότι πλέον το place αυτό έγινε approved ή disapproved αντίστοιχα.

```
102  // Approve Place
103  router.post('/approve/:placeId', (req, res) => {
104      const id = req.params.placeId
105      Place.updateOne({
106              _id: id
107          }, {
108              approved: true
109          })
110          .then(docs => {
111              res.redirect('/places/list')
112          })
113  })
114
115  // Disapprove Place
116  router.post('/disapprove/:placeId', (req, res) => {
117      const id = req.params.placeId
118      Place.updateOne({
119              _id: id
120          }, {
121              approved: false
122          })
123          .then(docs => {
124              res.redirect('/places/list')
125          })
126  })
```

Εικόνα 123: Ο κώδικας υλοποίησης των συναρτήσεων του place

Για να μπορέσει ένας χρήστης να δει τις πληροφορίες ενός συγκεκριμένου place χρειάζεται σύνδεση με τη βάση με πολλαπλά documents είναι απαραίτητες πληροφορίες που υπάρχουν τόσο στο place, όσο και στα user, booking και review καθώς όπως είδαμε στο front end κομμάτι στη σελίδα με τις πληροφορίες του place περιλαμβάνονται στοιχεία τα οποία βρίσκονται αποθηκευμένα σε όλα τα documents που αναφέρθηκαν.

```
128  // View Single Place
129  router.get('/view/:placeId', (req, res, next) => {
130      const id = req.params.placeId;
131      Place.findOne({
132          _id: id
133      })
134      .then(docs => {
135          var placeData = docs
136          const owner = docs.owner
137          User.findOne({
138              username: owner
139          })
140          .then(docs => {
141              const ownerData = docs
142              Booking.find({
143                  place: id
144              })
145              .sort('start_date')
146              .then(docs => {
147                  const bookingData = docs
148                  Review.find({
149                      place_id: id
150                  })
151                  .then(docs => {
152                      res.render('places/view', {
153                          placeData: placeData,
154                          ownerData: ownerData,
155                          bookingData: bookingData,
156                          reviewData: docs,
157                          user: req.user
158                      });
159                  })
160              })
161          })
162      })
```

Εικόνα 124: Ο κώδικας ανάκτησης πληροφοριών ενός συγκεκριμένου place

Για τη δημιουργία place χρησιμοποιείται στο πακέτο multer το οποίο καλείται από τη συνάρτηση upload.any() (γραμμή 198) και εκτελείται πριν εκτελεστεί ο υπόλοιπος κώδικας. Η συνάρτηση παίρνει την εικόνα που έχει ανεβάσει ο χρήστης στη φόρμα και την ανεβάζει στο server ούτως ώστε στη συνέχεια να αποθηκευτεί στη βάση. Στη συνέχεια δημιουργούνται όλες οι μεταβλητές που αντιστοιχούν στα στοιχεία που έχει εισάγει ο χρήστης στη φόρμα και αφού ελέγξει αν κάθε πεδίο έχει δεδομένα τότε δημιουργείται ένα νέο αντικείμενο τύπου place το οποίο στη συνέχεια αποθηκεύεται στη βάση.

```
// Create Place
router.post('/create', upload.any(), (req, res) => {
    let errors = [];
    // Check if image is uploaded
    if (req.files === undefined || req.files.length == 0) {
        res.redirect('/places/create');
    } else {
        const {
            name,
            description,
            owner = req.user.username,
            address,
            city,
            bedrooms,
            bathrooms,
            max_people,
            price,
            approved = false,
            features,
            image = req.files[0].path.slice(6)
        } = req.body;
        // Check if all fields are filled in
        if (!name || !description || !address || !city || !bedrooms || !bathrooms || !max_people || !price || !image) {
            errors.push({
                msg: 'Please enter all fields'
            });
        }
```

Εικόνα 125: Ο κώδικας δημιουργίας ενός place (1/2)

```
        // Redirect if errors exist
        if (errors.length > 0) {
            res.redirect('/places/list');
        } else {
            const newPlace = new Place({
                name,
                description,
                owner,
                address,
                city,
                bedrooms,
                bathrooms,
                max_people,
                price,
                approved,
                features,
                image
            });
            newPlace
                .save()
                .then(place => {
                    req.flash(
                        'success_msg',
                        'Place created!'
                    );
                    res.redirect('/places/list');
                })
                .catch(err => console.log(err));
        }
    }

});
```

Εικόνα 126: Ο κώδικας δημιουργίας ενός place (2/2)

Με παρόμοιο τρόπο όπως πραγματοποιήθηκαν οι παραπάνω λειτουργίες εκτελείται και η λειτουργία του **edit place** όπου όταν ένας χρήστης πραγματοποιεί αλλαγές στο place τότε γίνεται update στα ήδη αποθηκευμένα στοιχεία στη θέση των οποίων γράφονται τα καινούρια που εισάγει ο χρήστης.

Για τη λειτουργία **delete** όταν εκτελούνται τα queries στη βάση προκειμένου να βρεθούν τα δεδομένα που χρειάζεται να διαγραφούν γίνεται χρήση της remove προκειμένου να γίνει αφαίρεση αυτών.

```
341  // Delete Place
342  router.delete('/:placeId', (req, res) => {
343      const id = req.params.placeId;
344      Place.remove({
345          _id: id
346      })
347      .exec()
348      .then(result => {
349          // Delete bookings of deleted place
350          Booking.remove({
351              place: id
352          })
353          .exec()
354          .then(result => {
355              req.flash(
356                  'success_msg',
357                  'Place deleted!'
358              );
359              res.redirect('/places/list');
360          })
361      })
362      .catch(err => {
363          console.log(err);
364          res.status(500).json({
365              error: err
366          });
367      });
368  });
```

Εικόνα 127: Ο κώδικας διαγραφής ενός place

Προκειμένου να μην εμφανίζονται στον χρήστη οι κρατήσεις (booking) που έχει πραγματοποιήσει σε ένα place που διαγράφηκε πρέπει να διαγραφούν και τα αντίστοιχα booking που έγιναν για αυτό, για αυτό το λόγο πραγματοποιούνται και οι εντολές από τη γραμμή 350 έως 360.

Στη συνέχεια θα παρουσιαστούν κομμάτια από τα αρχεία routes που απευθύνονται στις υπόλοιπες βασικές οντότητες της εφαρμογής καθώς οι βασικές τεχνικές που ακολουθήθηκαν αναλύθηκαν ήδη για το place.

Bookings

```
7   // Create Booking
8   router.post('/:placeId', (req, res) => {
9       const place = req.params.placeId;
10      const user = String(req.user._id);
11      let errors = [];
12      if (!req.body.start || !req.body.end) {
13          errors.push({
14              msg: 'Please enter all fields'
15          });
16      }
17      if (errors.length > 0) {
18          res.redirect('/places/list')
19      } else {
20          const start_date = req.body.start
21          const end_date = req.body.end
22          const today = getCurrentDate()
23
24          // Check if dates are valid
25          if (start_date.localeCompare(end_date) <= 0 && start_date.localeCompare(today) >= 0 && end_date.localeCompare(today) >= 0) {
26              //Check if dates are free
27              Booking.find({
28                      place: place
29                  })
30                  .exec()
31                  .then(docs => {
32                      var free = true;
33                      for (i = 0; i < docs.length; i++) {
34                          if (!(end_date.localeCompare(docs[i].start_date) < 0 || start_date.localeCompare(docs[i].end_date) > 0)) {
35                              free = false
36                              break
37                          }
38                      }
```

Εικόνα 128: Ο κώδικας δημιουργίας ενός booking

Αρχικά δημιουργείται το router booking και στη συνέχεια γίνεται χρήση της function **localcompare** που αποτελεί default function την οποία μπορούν να καλέσουν μεταβλητές τύπου date όπως οι μεταβλητές start_date και end_date που δημιουργήθηκαν εδώ. Η συνάρτηση αυτή ελέγχει αν οι ημερομηνίες αυτές είναι σωστές π.χ αν έχει δηλωθεί από λάθος η ημερομηνία αποχώρησης νωρίτερα από την ημερομηνία άφιξης. Εφόσον δεν υπάρχει κάποιο πρόβλημα με τις ημερομηνίες τότε με την εντολή στη γραμμή 27 κάνει το αντίστοιχο query στη βάση για να ψάξει τις εγγραφές ούτως ώστε να βρει αν υπάρχει το αντίστοιχο place που ψάχνει ο χρήστης στη βάση.

```
101 // Extra Functions
102
103 function getCurrentDate() {
104     var today = new Date()
105     var dd = String(today.getDate()).padStart(2, '0')
106     var mm = String(today.getMonth() + 1).padStart(2, '0') // January is 0
107     var yyyy = today.getFullYear()
108
109     today = yyyy + '-' + mm + '-' + dd
110     return today
111 }
```

Εικόνα 129: Η συνάρτηση getCurrentDate

Με χρήση της συνάρτησης **getCurrentDate** παίρνουμε την μέρα του συστήματος ούτως ώστε να χρησιμοποιηθεί από την προηγούμενη συνάρτηση για να πραγματοποιηθούν σωστά οι έλεγχοι.

Παρομοίως δημιουργούνται και τα routes για το **City** και το **Review** όπου δεν κρίνεται σκόπιμο να αναλυθεί περαιτέρω κάποιο κομμάτι κώδικα.

Index

routes > JS index.js > ...

```
1   const express = require('express');
2   const router = express.Router();
3
4   const Place = require('../models/Place');
5   const Booking = require('../models/Booking');
6
7   // Load functions
8   const {
9     ensureAuthenticated,
10    forwardAuthenticated
11  } = require('../config/auth');
12
13
14  // Welcome Page
15  router.get('/', /*forwardAuthenticated,*/ (req, res) =>
16    res.render('index', {
17      user: req.user
18    })
19  );
20
```

Εικόνα 130: Ο κώδικας δημιουργίας της σελίδας index

Αρχικά γίνεται φόρτωση των μοντέλων και των function που θα χρειαστούν ενώ στη συνέχεια πραγματοποιείται η υλοποίηση του κώδικα που φορτώνει την αρχική σελίδα της εφαρμογής εάν ένας χρήστης δώσει το url της.

```
// Dashboard
router.get('/dashboard', ensureAuthenticated, (req, res, next) => {
  Place.find({
    owner: req.user.username
  })
  .exec()
  .then(docs => {
    const placeData = docs
    Booking.find({
      user: req.user._id
    })
    .exec()
    .then(docs => {
      res.render('dashboard', {
        placeData: placeData,
        bookingData: docs,
        user: req.user
      })
    })
  })
  .catch(err => {
    console.log(err);
    res.status(500).json({
      error: err
    });
  });
});
```

Εικόνα 131: Ο κώδικας δημιουργίας της σελίδας dashboard

Εδώ πραγματοποιείται ο κώδικας με βάση τον οποίων θα εμφανιστούν τα δεδομένα στο profile του χρήστη στην αντίστοιχη σελίδα της εφαρμογής. Πραγματοποιούνται δύο queries στη βάση που ψάχνουν τα place αλλά και τα booking που ανήκουν στον συγκεκριμένο χρήστη. Στη συνέχεια τα αποτελέσματα αυτών εμφανίζονται στον χρήστη μέσο της σελίδας profile της εφαρμογής.

Users

```
// Register
router.post('/register', (req, res) => {
  const { username, email, firstname, surname, password, password2 } = req.body;
  let errors = [];

  // Check for errors
  if (!username || !email || !firstname || !surname || !password || !password2) {
    errors.push({ msg: 'Please enter all fields' });
  }

  if (password != password2) {
    errors.push({ msg: 'Passwords do not match' });
  }

  if (password.length < 6) {
    errors.push({ msg: 'Password must be at least 6 characters' });
  }

  if (errors.length > 0) {
    // Go back and print errors
    res.render('register', {
      errors,
      username,
      email,
      firstname,
      surname,
      password,
      password2
    });
```

Εικόνα 132: Ο κώδικας των ελέγχων που γίνονται κατα το Registration

Εδώ χτίζεται ο κώδικας που πραγματοποιεί τους έλεγχους στα στοιχεία που θα δώσει ο χρήστης στη φόρμα register. Στη συνέχεια όπως θα δούμε ελέγχεται η βάση και εάν υπάρχει ήδη χρήστης που έχει καταχωρήσει το ίδιο mail ο χρήστης θα πάρει μήνυμα λάθους καθώς το mail υπάρχει ήδη και χρησιμοποιείται από άλλο χρήστη. Για λόγους ασφάλειας ο κωδικός που θα εισάγει ο χρήστης αποθηκεύεται κρυπτογραφημένος στη βάση δεδομένων με χρήση **salt** και **hash**, δυο πολυ γνωστές τεχνικές κρυπτογράφησης κωδικών.

```
46      User.findOne({ email: email }).then(user => {
47        if (user) {
48          errors.push({ msg: 'Email already exists' });
49          res.render('register', {
50            errors,
51            username,
52            email,
53            firstname,
54            surname,
55            password,
56            password2
57          });
58        } else {
59          const newUser = new User({
60            username,
61            email,
62            firstname,
63            surname,
64            password
65          });
66
67          // Salt and Hash password
68          bcrypt.genSalt(10, (err, salt) => {
69            bcrypt.hash(newUser.password, salt, (err, hash) => {
70              if (err) throw err;
71              newUser.password = hash;
72              newUser
73                .save()
74                .then(user => {
75                  req.flash(
76                    'success_msg',
77                    'You are now registered and can log in'
78                  );
79                  res.redirect('/users/login');
80                })
81                .catch(err => console.log(err));
82            });
```

Εικόνα 133: Η δημιουργία κρυπτογραφημένου κωδικού και νέου User

Για να πραγματοποιηθεί η λειτουργία login τρέχει η συνάρτηση authenticate στο πακέτο passport που είναι υπεύθυνη για τη διαχείριση του login, πραγματοποιεί τους ελέγχους που χρειάζονται ανάλογα με το αν τα στοιχεία είναι σωστά ή όχι ο χρήστης θα ανακατευθυνθεί στην αντίστοιχη σελίδα.

```
89  // Login
90  router.post('/login', (req, res, next) => {
91    passport.authenticate('local', {
92      successRedirect: '/dashboard',
93      failureRedirect: '/users/login',
94      failureFlash: true
95    })(req, res, next);
96  });
97
98  // Logout
99  router.get('/logout', (req, res) => {
100   req.logout();
101   req.flash('success_msg', 'You are logged out');
102   res.redirect('/users/login');
103 });
104
105 module.exports = router;
```

Εικόνα 134: Ο κώδικας υλοποίησης το Login και Logout

Views

Τα αρχεία που περιέχονται στα views είναι υπεύθυνα για την κατασκευή της μορφής που έχουν οι σελίδες της εφαρμογής.

Ο κώδικας δεν είναι καθαρά HTML κώδικας αλλά .ejs που περιέχουν και κώδικα JavaScript, απαραίτητο ούτως ώστε να γίνεται η μεταβίβαση δεδομένων από το back end στο front end.

views > layout.ejs > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5     <%- include('./partials/head'); %>
6   </head>
7
8   <body>
9     <%- include('./partials/header'); %>
10    <div class="container">
11      <div id="content-wrap">
12        <%- body %>
13      </div>
14      <%- include('./partials/footer'); %>
15    </div>
16  </body>
17
18  </html>
```

Εικόνα 135: Η βασική δομή κάθε σελίδας της εφαρμογής

Το **layout.ejs** είναι η διαρρύθμιση με βάση την οποία εμφανίζονται οι πληροφορίες στην οθόνη του χρήστη και χρησιμοποιείται για όλες τις σελίδες της εφαρμογής.

Η σελίδα **index.ejs** που είναι και η αρχική σελίδα της εφαρμογής εμφανίζει το logo της εφαρμογής στον χρήστη.

views > index.ejs > ...

```
1   <img class="logo" src="/images/logo.png" alt="My_logo">
```

Εικόνα 136: Ο κώδικας της σελίδας index.ejs

Η σελίδα **login.ejs** χρησιμοποιεί τα αρχεία partials το οποίο είναι υπεύθυνο για την εμφάνιση των μηνυμάτων που δημιουργούνται στα routes του login όπως είδαμε ανάλογα με την κατηγορία στην οποία θα πέσει ο χρήστης όταν προσπαθήσει να συνδεθεί στην εφαρμογή.

views > login.ejs > ...

```
<div class="row mt-5">
  <div class="col-md-6 m-auto">
    <div class="card card-body">
      <h1 class="text-center mb-3"><i class="fas fa-sign-in-alt"></i> Login</h1>
      <% include ./partials/messages %>
      <form action="/users/login" method="POST">
        <div class="form-group">
          <label for="email">Username</label>
          <input type="text" id="username" name="username" class="form-control" placeholder="Enter Username" />
        </div>
        <div class="form-group">
          <label for="password">Password</label>
          <input type="password" id="password" name="password" class="form-control" placeholder="Enter Password" />
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
      </form>
      <p class="lead mt-4">
        No Account? <a href="/users/register">Register</a>
      </p>
    </div>
  </div>
</div>
```

Εικόνα 137: Ο κώδικας της σελίδας login.ejs

Partials

Σε αυτά τα αρχεία βρίσκεται κώδικας που χρησιμοποιείται για να καλυφθούν λειτουργίες για τη δομή και τη μορφή όλων των αρχείων. Στο **head.ejs** υπάρχει ο κώδικας για τη μορφή τον οποίων μετά φορτώνει το layout. Περιλαμβάνει το CSS stylesheet, που είναι υπεύθυνο για τη μορφή, καθώς και τα μεταδεδομένα που χρησιμοποιούνται.

views > partials > head.ejs > ...

```
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta http-equiv="X-UA-Compatible" content="ie=edge" />
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="anonymous">
<link rel="stylesheet" href="https://bootswatch.com/4/journal/bootstrap.min.css" />
<link rel="stylesheet" href="/styles.css">
<link rel="icon" href="/images/logo.png" sizes="16x16" type="image/png">
<title>AirbNA</title>
```

Εικόνα 138: Ο κώδικας του αρχείου head.ejs

Το **header.ejs** περιέχει το navbar, που είναι το μενού επιλογών του χρήστη που εμφανίζεται σε όλες τις σελίδες της εφαρμογής και από αυτό μπορεί να επιλέξει ποιες ενέργειες θέλει να εκτελέσει. Χωρίζεται μπορούμε να πούμε σε δύο κομμάτια το αριστερό και το δεξί. Στο αριστερό κομμάτι έχουμε την εμφάνιση του logo που αποτελεί και link και οδηγεί στην αρχική σελίδα, και τις επιλογές create place και profile που οδηγούν στις αντίστοιχες σελίδες.

```
<nav class="navbar navbar-expand-lg">
    <a href="/">
        <img height="80%" width="50px" src="/images/logo.png">
    </a>
    <ul class="navbar-nav mr-auto">
        <li class="nav-item">
            <a class="nav-link" href="/">AirbNA</a>
        </li>
        <% if (typeof(user) !== 'undefined') { %>
        <li class="nav-item">
            <a class="nav-link" href="/places/create">Create Place</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/dashboard">Profile</a>
        </li>
        <% } %>
    </ul>
```

Εικόνα 139: Ο κώδικας δημιουργίας του αριστερού μέρους του navbar

```
    <ul class="navbar-nav ml-auto">
        <li class="nav-item">
            <form action="/places/results" method="POST">
                <input class="form-control" type="text" id="search" name="search" placeholder="Search"
                    aria-label="Search">
            </form>
        </li>
        <% if (typeof(user) == 'undefined') { %>
        <li class="nav-item">
            <a class="nav-link" href="/users/login">Login</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/users/register">Register</a>
        </li>
        <% } %>
        <% if (typeof(user) !== 'undefined') { %>
        <li class="nav-item">
            <a class="nav-link" href="/users/logout">Logout</a>
        </li>
        <% } %>
    </ul>
</nav>
```

Εικόνα 140: Ο κώδικας δημιουργίας του δεξιού μέρους του navbar

Το αρχείο **footer.ejs** περιέχει τα απαραίτητα script σε JavaScript που εμφανίζονται με κλήση από το layout σε όλες τις σελίδες της εφαρμογής.

views > partials > footer.ejs > ...

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous">
</script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js"
    integrity="sha384-wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut" crossorigin="anonymous">
</script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"
    integrity="sha384-B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k" crossorigin="anonymous">
</script>
```

Εικόνα 141: Ο κώδικας του αρχείου footer.ejs

Places

Για τη δημιουργία των σελίδων όπως αναφέρθηκε χρησιμοποιήθηκε τόσο κώδικας HTML όσο και αρχεία .ejs ενδεικτικά θα δοθούν κάποια παραδείγματα κώδικα για τη σελίδα του place καθώς όλες οι υπόλοιπες σελίδες βασίστηκαν στην ίδια τεχνική.

views > places > list.ejs > ...

```
1   <div class="row mt-5">
2     <div class="col-md-12 m-auto">
3       <div class="card card-body">
4         <h1 class="text-center mb-3">
5           <i class="fas fa-home"></i> View Places
6         </h1>
7         <% include ../partials/messages %>
8
9         <% if(placeData.length > 0) { %>
10        <table class="table-center">
11          <tr>
12            <th>Name</th>
13            <th>Description</th>
14            <th>Owner</th>
15            <th>Address</th>
16            <th>City</th>
17            <th>Bedrooms</th>
18            <th>Bathrooms</th>
19            <th>Max People</th>
20            <th>Price</th>
21          </tr>
22          <% for(var i=0; i < placeData.length; i++) { %>
23          <tr>
24            <td><a href="/places/view/<%= placeData[i]._id %>"><%= placeData[i].name %></a></td>
25            <td><%= placeData[i].description %></td>
26            <td><%= placeData[i].owner %></td>
27            <td><%= placeData[i].address %></td>
28            <td><%= placeData[i].city %></td>
29            <td><%= placeData[i].bedrooms %></td>
30            <td><%= placeData[i].bathrooms %></td>
31            <td><%= placeData[i].max_people %></td>
32            <td><%= placeData[i].price %></td>
33            <td>
34              <% if(placeData[i].approved == true) { %>
35              <form action="/places/disapprove/<%= placeData[i]._id %>" method="POST">
36                <button type="submit" class="btn btn-secondary btn-block">Disapprove</button>
37              </form>
```

Εικόνα 142: Ο κώδικας εμφάνισης της σελίδας View Places

Όπως αναφέρθηκε γίνεται include των partials στη συνέχεια κατασκευάζεται ο πίνακας με τα στοιχεία που θα δει ο χρήστης που παίρνουν τιμές από τα αντίστοιχα στοιχεία της βάσης. Στη γραμμή 34 βλέπουμε ότι γίνεται έλεγχος και εφόσον το place είναι approved εμφανίζεται στον χρήστη διαφορετικά δεν θα πάρει κάποιο αποτέλεσμα στην οθόνη του.

Στη συνέχεια βλέπουμε τον κώδικα που κατασκευάζει τα κουμπιά της φόρμας και ορίζεται το που θα κατευθυνθεί ο χρήστης μετά την χρήση αυτών.

```
38                <% } else { %>
39                <form action="/places/approve/<%= placeData[i]._id %>" method="POST">
40                  <button type="submit" class="btn btn-primary btn-block">Approve</button>
41                </form>
42                <% } %>
43              </td>
44              <td>
45                <form action="/places/edit/<%= placeData[i]._id %>" method="GET">
46                  <button type="submit" class="btn btn-primary btn-block">Edit</button>
47                </form>
48              </td>
49              <td>
50                <form action="/places/<%= placeData[i]._id %>?_method=DELETE" method="POST">
51                  <button type="submit" class="btn btn-danger btn-block">Delete</button>
52                </form>
53              </td>
54            </tr>
55            <% } %>
56          </table>
57          <% } else { %>
58          No place data to display.<br><br>
59          <% } %>
60          <br>
61          <p class="table-buttons">
62            <a class="btn btn-primary" href="/places/create">Create</a>
63            <a class="btn btn-secondary" href="/dashboard">Back</a>
64          </p>
65        </div>
66      </div>
67    </div>
```

Εικόνα 143: Η συνέχεια του κώδικα του View Places

7 Επίλογος

7.1 Σύνοψη και Συμπεράσματα

Στη διπλωματική μελετήθηκαν οι βάσεις δεδομένων MySQL και MongoDB καθώς και εφαρμογές εύρεσης καταλυμάτων ενώ αναπτύχθηκαν τρεις διαφορετικές τέτοιου τύπου εφαρμογές με χρήση των βάσεων που αναφέρθηκαν. Βασική ιδέα είναι να αποφασιστεί με βάση την κατασκευή των εφαρμογών αυτών αλλά και τη χρήση τους ποια από τις βάσεις δεδομένων αυτές είναι περισσότερο φιλικές προς τον δημιουργό των εφαρμογών. Τα συμπεράσματα που προκύπτουν είναι ότι δεν υπάρχει ξεκάθαρη απάντηση καθώς κάθε τεχνική προσφέρει τα δικά της πλεονεκτήματα και μειονεκτήματα.

Η εφαρμογή που αναπτύχθηκε με κώδικα γραμμένο σε PHP και HTML και βασίστηκε σε βάση MySQL παρουσιάζει το πλεονέκτημα ότι τα δεδομένα που αποθηκεύονται στη βάση έχουν προκαθορισμένο σχήμα και είναι διαχωρισμένα σε οντότητες από τον προγραμματιστή οπότε είναι και περισσότερο εύκολο να διαχωριστεί το ποια πληροφορία αντιστοιχεί στην κάθε οντότητα και τις σχέσεις μεταξύ αυτών παρόλα αυτά θα μπορούσε να θεωρηθεί και ως μειονέκτημα καθώς παρουσιάστηκαν προβλήματα στη διαχείριση των δεδομένων αυτών ειδικά όσο αυξάνονταν τα δεδομένα που είναι αποθηκευμένα στη βάση. Η εφαρμογή με χρήση του μοντέλου Wordpress παρουσιάζει το πλεονέκτημα της αυτόματης δημιουργίας της βάσης και ο προγραμματιστής δεν χρειάζεται να συμμετάσχει καθόλου στο κομμάτι αυτό ωστόσο παρουσιάζεται το μειονέκτημα της τροποποίησης στοιχείων αυτής και συντήρησης της αφού ο προγραμματιστής δεν γνωρίζει τη δομή αυτής ενώ δεν προσφέρει δυνατότητες προσαρμογής ούτως ώστε να μπορέσει να φιλοξενήσει μεγάλο όγκο δεδομένων. Τέλος η εφαρμογή που δημιουργήθηκε με χρήση Node.js και χρησιμοποίησε βάση τύπου MongoDB δίνει τη δυνατότητα διαχείρισης πολλαπλών δεδομένων πολύ πιο γρήγορα συγκριτικά με τις υπόλοιπες βάσεις ωστόσο η δομή αυτής αποτελεί μία πιο πολύπλοκη έννοια σε σύγκριση με τα δεδομένα που πήραμε από τη χρήση των υπόλοιπων βάσεων. Επίσης με βάση την έρευνα που πραγματοποιήθηκε σε παρόμοιες εφαρμογές του εμπορίου όπως η Airbnb που είναι και η πιο διαδεδομένη βλέπουμε ότι ενώ γίνεται χρήση της MySQL πραγματοποιούνται τροποποιήσεις πάνω σε αυτή καθώς από μόνη της δεν είναι κατάλληλη για τον όγκο δεδομένων που απαιτείται από τις εγαρμογές αυτές και παρουσιάζονται προβλήματα.

7.2 Μελλοντικές επεκτάσεις

Για να δοθεί μία πιο σαφής απάντηση ως την καταλληλότερη βάση για εφαρμογή τύπου εύρεσης καταλύματος θα μπορούσε κάποιος να πραγματοποιήσει τροποποιήσεις στις εφαρμογές που ήδη δημιουργήθηκαν προσθέτοντας δεδομένα στα ήδη υπάρχοντα συστήματα ούτως ώστε να αντληθούν και συμπεράσματα ως προς την ταχύτητα κάθε βάσης στην εκτέλεση ερωτημάτων ή ακόμα και να δημιουργηθούν περαιτέρω εφαρμογές με χρήση διαφορετικών βάσεων που θα μπορούσαν να κριθούν καταλληλότερες έπειτα από σύγκριση με τις ήδη υπάρχουσες.

8 Βιβλιογραφία

1. Σκουρλάς, Χ., (2001), *Υλοποίηση Εφαρμογών με Γλώσσα SQL: Χρήση Τεχνολογίας Oracle και Developer/2000*, Εκδόσεις Νέων Τεχνολογιών
2. Ullman, J. D., & Widom, J., Μετάφραση: Βερύκιος Β., (2008), *Βασικές Αρχές για τα Συστήματα Βάσεων Δεδομένων*, Εκδόσεις Κλειδάριθμος
3. Ramakrishnan, R. & Gehrke, J., (2002), *Database Management Systems*, McGraw-Hill
4. Bassil, Y., (2011), *A Comparative Study on the Performance of the Top DBMS Systems*, Journal of Computer Science & Research (JCSCR), 1 (1) 20-31, Αναρτήθηκε Φεβρουάριος 2012, από https://www.scienceopen.com/document?vid=36932026-ded1-4dd8-81aa-cb810c704ca9
5. Truică, C. O., Rădulescu, F., Boicea, A. & Bucur, I., (2018), *Performance evaluation for CRUD operations in asynchronously replicated document oriented database*, από https://www.scienceopen.com/document?vid=155f2900-ccb1-46c3-b24f-2cafd13be3d7
6. Győrödi, C., Győrödi, R., Pecherle, G. & Olah, A., (2015), *A comparative study: MongoDB vs. MySQL*, International Conference on Engineering of Modern Electric Systems (EMES), Αναρτήθηκε 12 Ιουνίου 2015, από https://ieeexplore.ieee.org/abstract/document/7158433
7. *database (DB)*, (Ιούλιος 2019), SearchSQLServer.com, από https://searchsqlserver.techtarget.com/definition/database
8. *NoSQL (Not Only SQL database)*, (Δεκέμβριος 2020), SearchSQLServer.com, από https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL
9. *Why MySQL?*, mysql.com, από https://www.mysql.com/why-mysql/
10. *Types of Databases*, javatpoint.com, από https://www.javatpoint.com/types-of-databases
11. *MySQL 5.7 Reference Manual*, (26 Δεκεμβρίου 2020), mysql.com, από https://dev.mysql.com/doc/refman/5.7/en/
12. *Trivago*, από https://www.trivago.gr
13. *Airbnb*, από https://www.airbnb.gr
14. *Booking.com*, από https://booking.kayak.com
15. *Τι είναι τα CMS (π.χ. Joomla, Wordpress, Drupal)?*, ip.gr, από https://www.ip.gr/Web_Development/τι-είναι-τα-cms-joomla-wordpress-drupal-246.html
16. *What is PHP?*, php.net από https://www.php.net/manual/en/intro-whatis.php
17. *front end and back end*, (Μαϊος 2019), whatis.com, από https://whatis.techtarget.com/definition/front-end

18. *CSS Overview*, getbootstrap, από https://getbootstrap.com/docs/3.4/css/
19. *SQL FOREIGN KEY Constraint*, w3schools, από https://www.w3schools.com/sql/sql_foreignkey.asp
20. *What is JavaScript used for?*, (18 Οκτωβρίου 2018), hackreactor, από https://www.hackreactor.com/blog/what-is-javascript-used-for
21. *How a web session is defined in Universal Analytics*, google.com, από https://support.google.com/analytics/answer/2731565?hl=en
22. *Wordpress Features Overview*, Wordpress, από https://wordpress.com/features/
23. *Bringing MySQL to the web*, phpmyadmin, από https://www.phpmyadmin.net
24. *What Is MongoDB?*, mongodb, από https://www.mongodb.com/what-is-mongodb
25. *Login Logout Menu*, wordpress, από https://wordpress.org/plugins/baw-login-logout-menu/
26. *Duplicate Page*, wordpress, από https://wordpress.org/plugins/duplicate-page/
27. *Elementor Website Builder*, elementor, από https://elementor.com
28. *Contact Form, Drag and Drop Form Builder for WordPress – Everest Forms*, wordpress, από https://wordpress.org/plugins/everest-forms/
29. *WP Hotel Booking*, wordpress, από https://wordpress.org/plugins/wp-hotel-booking/
30. *Customize Your Login Page with the Memphis Custom Login Plugin*, managewp, από https://managewp.com/blog/memphis-custom-login-plugin
31. *Say What?*, wordpress, από https://wordpress.org/plugins/say-what/
32. *User Role Editor*, wordpress, από https://wordpress.org/plugins/user-role-editor/